

APPLICATION OF GEOMETRIC PROGRAMMING
TO INDUSTRIAL SYSTEMS

by 1264

NAYAN BHATTACHARYA

B. Tech. (Hons.), Mechanical, Indian Institute of Technology
Kharagpur, India, 1966

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree


MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1969

Approved by:


Major Professor

LD
2668
T4
1969
B48

ACKNOWLEDGEMENT

The author wishes to express his deep sense of appreciation to his major professor, Dr. E. S. Lee for his guidance, constructive criticism, helpful suggestions and the personal interest taken in the preparation of this thesis.

TABLE OF CONTENTS

CHAPTER		Page
I.	INTRODUCTION	1
II.	GEOMETRIC PROGRAMMING	4
	Extension of Geometric Programming . . .	6
	Computational Procedure	10
III.	A LAGRANGIAN ALGORITHM	14
IV.	APPROXIMATION TECHNIQUES	18
	Example 1	18
	Example 2	18
	Example 3	19
	Example 4	20
	Example 5	21
	Example 6	21
V.	APPLICATIONS	23
	A Sample Problem	23
	Sea Power - Heat Exchanger Problem . . .	26
	Condenser Problem	40
	Chemical Equilibrium Problem	49
	Transformer Problem	56
	Production Inventory Problem	61
VI.	DISCUSSION	73
	REFERENCES	75
	APPENDIX A: FLOW CHARTS	77
	APPENDIX B: COMPUTER PROGRAMS	82

LIST OF TABLES

Table	Page
1. Convergence rate of the sample problem by the Hooke and Jeeves method	25
2. Convergence rate of the sea power problem by the Newton-Raphson method	35
3. Convergence rate of the sea power problem by the Hooke and Jeeves method	36
4. Optimum values of design parameters of the sea-power problem	37
5. Convergence rate of the condenser design problem by the Hooke and Jeeves method	48
6. Convergence rate of the chemical equilibrium problem by the Hooke and Jeeves method	55
7. Convergence rate of the transformer problem by the Hooke and Jeeves method	58
8. Convergence rate of the transformer problem by the Newton-Raphson method	59
8a. Computational aspects of problems 1 to 5	60
9. Typical convergence rate of the inventory problem with starting values $x_i = 10$, $i = 1, \dots, 10$	67
10. Typical convergence rate for the inventory problem with starting values $x_i = 5.0$, $i = 1, \dots, 10$	68
11. Effect of forcing on convergence	69
12. Optimum inventory and production levels	70

LIST OF FIGURES

Figure		Page
1.	Temperature distribution in the sea power problem	30
2.	The convergence rate of the sea power problem by the Newton-Raphson method	38
3.	The rate of change of variables in the sea power problem by the Newton-Raphson method . .	39
4.	The convergence rate of the condenser problem by the Hooke and Jeeves method	47
5.	The convergence rate of the chemical equilibrium problem by the Hooke and Jeeves method	54
6.	The convergence rate of the transformer problem by the Newton-Raphson method	57
7.	Optimum inventory levels	71
8.	Optimum production levels	72
9.	Flow diagram for geometric programming	78
10.	Flow diagram for Hooke and Jeeves pattern Search	79
11.	Flow diagram for exploratory moves	80
12.	Flow diagram for the Lagrangian algorithm . .	81

I. INTRODUCTION

The increased use of mathematical models in the analysis and optimization of industrial systems is one of the significant developments of modern engineering practice. Optimal desing of process equipment often involves finding numerical values for the design parameters to minimize a cost function, usually nonlinear, subject to design constraints. Most of the models which accurately describe the real-life systems prove to be too complex for solution by available algorithms. This is especially true of problems in which the constraints are nonlinear.

Recently, the geometric programming technique, has been developed which can handle a subclass of the above problems in which the cost function and the constraints are generalized polynomials.

In 1961 Zener [19] observed that the sum of the component costs sometimes may be minimized almost by inspection when each cost depends only on the products of the design variables, each raised to an arbitrary but known power. Duffin and Peterson [6] extended Zener's work. Zener and his associates' work had been restricted to functions they called 'Posynomials,' which are generalized polynomials with positive coefficients. Passy and Wilde [12] further generalized the method to include negative coefficients and reversed inequalities.

Geometric programming is specially suitable for engineering optimization problems based on desing relations developed either by dimensional analysis or by fitting power functions to

experimental results.

An important feature of geometric programming is its computational convenience. When the number of terms exceeds the number of variables by a small number, the computations are much simpler than the highly nonlinear character of the problem would lead one to expect. To minimize an unconstrained polynomial of m variables, the conventional method of calculus involves the solution of m nonlinear equations. On the other hand, if the function to be minimized contains exactly $m + 1$ terms, the problem can be solved by geometric programming by solving $m + 1$ linear equations, a far easier task. This is advantageous when the problem involve inequality constraints.

Although Passy and Wilde [12] have extended the geometric programming algorithm to handle objective functions and constraints with negative coefficients, difficulty is often encountered in numerical analysis except in the special case where there is exactly one more term than there are independent variables.

Recently Blau and Wilde [5] developed a Lagrangian algorithm for generalized polynomial optimization with equality constraints. The method optimizes the Lagrangian function with the Newton-Raphson procedure. This algorithm can handle negative coefficients efficiently and converges rapidly. One difficulty with this method is its occasional use of too large a step, which prevents convergence. This difficulty was overcome in this work by using a forcing procedure which restricts the maximum step size to a predetermined percentage of the variables.

The purpose of this thesis is to apply geometric programming

to different engineering design and industrial management systems in production planning and to analyze geometric programming's merits and faults. In the following chapter the basic algorithm of geometric programming with extensions and the algorithm of Lagrangian polynomial optimization technique are discussed. A brief review of computational procedure and approximation technique follows. In Chapter V, various possible fields of application of the above algorithms are analyzed and finally the advantages and disadvantages of geometric programming are highlighted.

II. GEOMETRIC PROGRAMMING

The theory of geometric programming is based on the arithmetic-geometric mean inequality. The set of functions comprising the mathematical model, when expressed in terms of the primal variables, is called the primal problem. A dual formulation of the primal problem can be obtained. Minimization of the primal problem is equivalent to maximization of the dual problem and the two extreme values are equal.

In this chapter only the algorithms of geometric programming and its extensions are stated and the computational procedures for them are discussed. A detailed derivation of the algorithm and the proof of the theory can be found in [6] and [18].

A set of $p + 1$ generalized polynomials consisting of m real positive variables x_j can be expressed as:

$$g_k = \sum_{i \in J(k)} C_i \prod_{j=1}^m A_{ij} x_j \quad (1)$$

where $k = 0, 1, \dots, P$

and $J(k)$ is a set of integers ranging from m_k to n_k , thus:

$$J(k) = \{m_k, m_k + 1, \dots, n_k\} \quad (2)$$

$$\text{and } m_0 = 1, m_1 = n_0 + 1, \dots, m_p = n_p + 1, n_p = n \quad (3)$$

$$C_i > 0 \quad (4)$$

$$x_j > 0 \quad (5)$$

n is the total number of terms in the set of polynomials.

A_{ij} are any real numbers.

The primal problem is to minimize

$$g \equiv g_0$$

(6)

subject to the constraints

$$g_k \leq 1; \quad k = 1, 2, \dots, P$$

(7)

The associated dual problem can be formed consisting of a set of n dual variables δ_i satisfying a normality condition

$$\sum_{i \in J(0)} \delta_i = 1$$

(8)

and m orthogonality conditions:

$$\sum_{i=1}^n A_{ij} \delta_i = 0 \quad j = 1, 2, \dots, m$$

(9)

as well as n non-negativity conditions

$$\delta_i \geq 0 \quad i = 1, 2, \dots, n$$

(10)

The corresponding dual problem can be written as

$$V(\delta) = \left(\begin{array}{c} n \\ \prod_{i=1}^n \left(\frac{C_i}{\delta_i} \right) \delta_i \end{array} \right) \quad \begin{array}{c} P \\ \prod_{k=1}^P \lambda_k \end{array} \quad \lambda_k$$

(11)

$$\text{where } \lambda_k = \sum_{i \in J(k)} \delta_i \quad k = 1, 2, \dots, P$$

(12)

The logarithm of the dual function (11) is strictly concave and hence it has only one stationary point - a global maximum. So the minimum of g_0 is obtained by maximizing the dual function (11) subject to the normality and orthogonality conditions (8)

to (10).

Once the dual variables δ_i are known, the corresponding values of the primal variables x_j are found from the following relations:

$$C_i \prod_{j=1}^m x_j^{A_{ij}} = \delta_i g_o^* \quad (13)$$

for $i \in J(0)$

and

$$C_i \prod_{j=1}^m x_j^{A_{ij}} = \delta_i / \lambda_k \quad (14)$$

for $i \in J(k)$
 $k \neq 0$

where g_o^* = minimum value of the objective function.

EXTENSION OF GEOMETRIC PROGRAMMING

The following algorithm is obtained by Passy and Wilde (12); it extends the theory of geometric programming to take into account negative coefficients and reversed inequalities.

$P + 1$ generalized polynomial functions $g_k(x)$ can be expressed

as

$$g_k = \sum_{t=1}^{T_k} \sigma_{kt} C_{kt} \prod_{j=1}^m x_j^{A_{ktj}} \quad k = 0, 1, \dots, P \quad (15)$$

where

$$\sigma_{kt} = \pm 1 \quad (16)$$

$$C_{kt} > 0 \quad (17)$$

$$x_j > 0 \quad (18)$$

and A_{ktj} are real numbers. The signum functions σ_{kt} , the coefficients C_{kt} , T_k (the number of terms in g_k), and the A_{ktj} are all given. Then the typical optimization problem can be written as

$$\min g_0(x) = g_0(x^*) \equiv g_0^* \neq 0, \pm \infty \quad (19)$$

(* corresponds to the optimal solution).

subject to P inequality constraints.

$$0 < \sigma_k g_k^{\sigma_k} \leq 1 \quad k = 1, \dots, P \quad (20)$$

where σ_k are known signum functions.

This problem can be solved by working with a set of real finite dual variables δ_{kt} , one for each term of the g_k , which satisfies the following nonnegativity condition

$$\delta_{kt} \geq 0 \quad \text{for all } k \text{ and } t \quad (21)$$

and the normality condition:

$$\delta_{00} = \sigma_0 \sum_{t=1}^{T_0} \sigma_{0t} \delta_{0t} = 1 \quad (22)$$

the m orthogonality conditions:

$$\sum_{k=0}^P \sum_{t=1}^{T_k} \sigma_{kt} A_{ktj} \delta_{kt} = 0 \quad j = 1, \dots, m \quad (23)$$

and P inequality constraints:

$$\delta_{ko} = \sigma_k \sum_{t=1}^{T_k} \sigma_{kt} \delta_{kt} \geq 0 \quad k = 1, \dots, P \quad (24)$$

with the qualification that

$$\delta_{kt} = 0 \quad (25)$$

if and only if

$$\delta_{ko} = 0 \quad k = 1, \dots, P \quad (26)$$

σ_o must be chosen to satisfy the constraints.

The dual function can be written as

$$V(\delta, \sigma_o) = \sigma_o \left[\prod_{k=0}^P \prod_{t=1}^{T_k} \left(\frac{C_{kt} \delta_{ko}}{\delta_{kt}} \right)^{\sigma_{kt} \delta_{kt}} \right]^{\sigma_o} \quad (27)$$

with the assumption that

$$\lim_{\delta_{kt} \rightarrow 0} \left(\frac{C_{kt} \delta_{ko}}{\delta_{kt}} \right)^{\sigma_{kt} \delta_{kt}} = 1 \quad (28)$$

When all signum functions are not positive $g_o(x)$ is not, in general, convex and may have several constrained local minima, maxima or saddle points, and no simple duality relation holds. It is proved instead that to each critical point (called a pseudominimum) x^o of g_o , there corresponds a dual point (δ^o, σ_o) where V is a pseudomaximum and such that

$$g_o(x^o) = V(\delta^o, \sigma_o) \quad (29)$$

Roughly speaking, a pseudominimum is a point where g_o satisfies the Kuhn-Tucker constraint qualification as well as the differential form of the Kuhn-Tucker necessary conditions for a constrained local minimum.

$$\text{Then } g_o(x^o) = \text{Pmin } g_o(x) = \text{Pmax } V(\delta^o, \sigma_o) = V(\delta_o, \sigma_o) \quad (30)$$

where Pmin is an abbreviation for pseudominimum.

Then at a global minimum:

$$\text{Min } g_o(x^*) = \text{Min } [\text{Pmax } V(\delta^*, \sigma_o)] \quad (31)$$

Once the dual variables δ^* are known the primal variables are found from the following relations:

$$C_{ot} \prod_{j=1}^m x_j^{\Lambda_{otj}} = \delta_{ot} \sigma_o g_o^* \quad t = 1, \dots, T_o \quad (32)$$

and

$$C_{kt} \prod_{j=1}^m x_j^{\Lambda_{ktj}} = \frac{\delta_{kt}}{\delta_{ko}} \quad t = 1, \dots, T_m \quad (33)$$

$k = 1, \dots, P$

From equation (32), it can be seen that σ_o will have the same sign as g^o . Since there will always be more terms than variable, x_j , m equations can be found which are solvable for m primals. The solution of these equations is not difficult since they are linear in $\log x_j$.

(* corresponds to optimal solution).

COMPUTATIONAL PROCEDURE

The detailed computer flow charts and programs are provided in the appendix. The present discussion on the method is to aid the understanding of the subsequent discussions.

The computer algorithm finds the minimum of the primal function (6) subject to primal constraints (7) by maximizing the dual function (11) subject to dual constraints (8) through (10). Having found this maximum a transformation is made to obtain the primal variables x_j .

As can be seen the dual problem has n variables and $m + 1$ linear equality constraints. This gives the problem $n - (m + 1)$ degrees of freedom. Zener and his associates call this the degree of difficulty.

The dual problem with nonlinear objective function and linear equality constraints can be maximized by any conventional method, such as, by Lagrange multipliers or the gradient projection method. As suggested by Duffin [6] the dual function can be transformed to eliminate the linear equalities to result in a 'd' dimensional optimization problem, where 'd' is the degree of freedom. The transformation is done as follows:

The dual variables $\underline{\delta}$ satisfying the equations (8) and (9) can be written as sum of a normality and a set of nullity vectors by the method of linear algebra.

$$\text{or } \underline{\delta} = \underline{b}_0 + \sum_{j=1}^d r_j \underline{b}_j \quad (34)$$

where \underline{b}_0 = normality vector

\underline{b}_j = nullity vectors $j = 1, \dots, d$.

r_j = are arbitrary real numbers

satisfying the positivity constraints

$$b_i^0 + \sum_{j=1}^d r_j b_i^j \geq 0 \quad i = 1, \dots, n \quad (35)$$

writing the dual function (11) in transformed form

$$V(r) = k_0 \left(\begin{array}{c} d \\ \pi \\ j=1 \end{array} \begin{array}{c} r_j \\ k_j \end{array} \right) \left(\begin{array}{c} n \\ \pi \delta_i(r) \\ i=1 \end{array} \begin{array}{c} -\delta_i(r) \\ \lambda_k \end{array} \right) \begin{array}{c} P \\ \pi \lambda_k \\ k=1 \end{array} \quad (36)$$

where

$$k_j = \begin{array}{c} n \\ \pi C_i \\ i=1 \end{array} b_i^j \quad j = 0, 1, \dots, d \quad (37)$$

$$\delta_i(r) = b_i^0 + \sum_{j=1}^d r_j b_i^j \quad i = 1, \dots, n \quad (38)$$

This function can be maximized with respect to r_j by any direct search technique. It has been found that Hooke and Jeeves (10) direct search is quite efficient. The first four problems in chapter V have been solved by this method. Another approach is to obtain a set of 'd' equations by differentiating this function with respect to r_j and setting the result equal to zero:

$$k_j - \left(\begin{array}{c} d \\ \pi \delta_i \\ i=1 \end{array} \begin{array}{c} b_i^j \\ \lambda_k \end{array} \right) \begin{array}{c} P \\ \pi \lambda_k \\ k=1 \end{array} = 0 \quad j = 1, \dots, d \quad (39)$$

where

$$\lambda_k^j = \sum_{j \in J(k)} r_j b_i^j \quad j = 1, \dots, d \quad (40)$$

These sets of equations can be solved by the Newton-Raphson method to give optimum values of r_j and hence the dual variables.

Convergence by this method is not guaranteed, but when the method works the function converges very rapidly. Problems 2 and 5 are solved by this method.

The computational procedure can be briefly summarized as follows:

a) Mathematical formulation of the dual problem.

To obtain the dual problem in the form of equation (11), the primal problem and the constraints have to be in the form of equations (6) and (7). Many problems which are not in this form can be transformed into the required form by various techniques discussed in a later chapter.

b) Calculation of normality and nullity vectors.

The normality and nullity vectors of the form of equation (34) can be obtained by the usual method of linear algebra (9).

c) Obtaining the initial feasible solution of r_j .

The nonnegativity constraints (35) have to be satisfied for the initial feasible solution. This is achieved by adjusting all variables simultaneously.

d) Optimization of the dual function:

Any suitable method for optimization is applicable. Hooke and Jeeves pattern search [10] is mostly used in the problems that are solved in this work. Differentiation and the Newton-Raphson method [14] are also used where the function converges.

Since $V(\delta)$ is only defined for $\delta_1 \geq 0$ any intermediate step in the search procedure not meeting this condition is avoided.

The accuracy obtained in the solutions depends on the compromise between improvement in objective function and required computer time. Different accuracies ϵ , (ranging from .01 to .0001) are assumed for different problems. The function is assumed to converge when the function value changes by ϵ or less in two successive iterations of Newton-Raphson or step size is equal to or smaller than ϵ in Hooke and Jeeve search.

e) Transformation from dual to primal problem:

The primal variables are calculated from the optimum dual variables by equation (13) and (14).

The above procedure is effective for functions with positive coefficients. For functions with negative coefficients and reversed inequalities along positivity conditions (21), P inequality constraints (24) are to be satisfied and the minimum of all the pseudomaxima of the dual function (27) is to be found. It was found that the above procedure fails in this case because of limitations of search procedures. An efficient algorithm has been presented by Blau and Wilde [3] to handle polynomials with negative coefficients and with equality constraints. This algorithm is presented in the next chapter.

III. A LAGRANGIAN ALGORITHM.

In this chapter an algorithm for optimization of generalized polynomials with equality constraints is presented. This algorithm is of high practical importance since it is the only algorithm which handles negative coefficients in the polynomials effectively; most physical restrictions occurring in practice are often strict equalities.

The basic idea is to use a Newton-Raphson procedure [18] to drive to zero the components of the gradient of a Lagrangian function formed from the logarithms of the original objective function and the constraints. A nonlinear transformation, which amounts to substituting a weighting variable for each term, makes the Lagrangian gradient linear in the weights as well as in the Lagrange multipliers.

No proof for local convergence is yet available. For justification of the procedure the reader is referred to [3].

A set of $M + 1$ generalized polynomials of N variables x_n g_m can be defined as:

$$g_m = \sum_{t=1}^{T_m} \sigma_{mt} C_{mt} \prod_{n=1}^N x_n^{A_{mntn}} \quad m = 0, 1, \dots, M \quad (1)$$

$$\text{where } 0 < x_n < \infty \quad (2)$$

$$\text{and } C_{mt} > 0 \quad t = 1, \dots, T_m \quad (3)$$

$$\sigma_{mt} = \pm 1 \quad (4)$$

A_{mntn} is any real number

The minimization problem can be stated as

$$\min_x g_0 \quad (5)$$

$$\text{subject to } g_m = 1 \quad (m \neq 0) \quad (6)$$

To initiate the algorithm, finite positive values x_n^0 of x_n have to be chosen not necessarily satisfying the constraints (6). The initial value of the objective function is calculated as g_0^0 and the initial weights as

$$W_{mt}^0 = \begin{cases} (g_0^0)^{-1} C_{ot} \prod_{n=1}^N (x_n^0)^{A_{otn}} & \text{for } m = 0 \\ C_{mt} \prod_{n=1}^N (x_n^0)^{A_{mnt}} & \text{for } m = 1, \dots, M \end{cases} \quad (7)$$

At the i th iteration the following sums are calculated as

$$S_{mn}^i = \sum_{t=1}^{T_m} \sigma_{mt} A_{mnt} W_{mt}^i \quad (8)$$

From them the $N \times 1$ dimensional vector is formed as

$$\underline{S}_0^i = (S_{10}^i \dots S_{M0}^i)^T \quad (9)$$

and the $N \times M$ matrix

$$S^i = \begin{bmatrix} S_{11}^i & \dots & S_{1M}^i \\ S_{N1}^i & \dots & S_{NM}^i \end{bmatrix} \quad (10)$$

This matrix is assumed to have rank M , so that $\begin{bmatrix} S^{iT} & S^i \end{bmatrix}$ is non-singular. Then the initial $M \times 1$ vector of multipliers can be computed as

$$\underline{\lambda}^0 = (S^{0T} S^0)^{-1} S^{0T} S^0 \quad (11)$$

A $N \times N$ symmetric matrix T^1 can be computed as

$$\begin{aligned} T_{nj}^1 = & - \sum_{t=1}^{T_0} \sigma_{ot} A_{otn} A_{otj} W_{ot}^1 \\ & + \sum_{m=1}^M \lambda_m^1 \sum_{t=1}^{T_m} \sigma_{mt} A_{mtm} A_{mtj} W_{mi}^1 \end{aligned} \quad (12)$$

At each iteration there is a value of one additional variable v^1 which is also adjusted by the algorithm. To begin with, it is taken as the value of the objective function at x^0 .

$$\text{Let } \sigma^1 = g_o^1 / |g_o^1|$$

Then the $(M + N + 1)^2$ symmetric Newton-Raphson matrix R^1 is assembled as

$$R^1 = \begin{bmatrix} T^1 & S_o^1 & S^1 \\ S_o^{iT} & -1 & \underline{0} \\ S^{iT} & \underline{0} & \underline{0} \end{bmatrix} \quad (13)$$

where $\underline{0}$ represents the null matrices with appropriate dimensions.

Next a $(M + N + 1) \times 1$ dimensional error vector e^1 is formed as

$$e^i = \begin{bmatrix} S_o^i - S_{\lambda}^i \\ 1 - g_o^i (V^i)^{-1} \\ 1 - g_1^i \\ 1 - g_m^i \end{bmatrix} \quad (14)$$

Then the $(N + M + 1) \times 1$ dimensional correction vector is given by

$$\begin{bmatrix} \Delta \ln \underline{X}^i \\ \Delta \ln \sigma^i V^i \\ \Delta \underline{\lambda}^i \end{bmatrix} = (R^i)^{-1} e^i \quad (15)$$

so the next estimate of x^{i+1} is

$$x_n^{i+1} = X_n^i \exp (\ln X_n^i) \quad (16)$$

whereas

$$V^{i+1} = V^i \exp (\ln \sigma^i V^i) \quad (17)$$

$$\underline{\lambda}^{i+1} = \underline{\lambda}^i + \Delta \underline{\lambda}^i \quad (18)$$

These quantities are used to compute the new values of weights defined in equation (7) for $m = 1, \dots, M$.

For $m = 0$, the following equation is used.

$$W_{ot}^{i+1} = (V^{i+1})^{-1} C_{ot} \sum_{n=1}^N X_n^{A_{otn}} \quad (19)$$

Thus the algorithm completes the i th iteration. The procedure continues until all components of the error vector are acceptably close to zero.

IV. APPROXIMATION TECHNIQUES

Many optimization problems can be transformed into standard geometric programming problems, even though they are not explicitly expressed in posynomial form. This fact is illustrated in the following examples.

Example 1.

Minimize the function

$$G(\underline{x}) = f(\underline{x}) + [q(\underline{x})]^a h(\underline{x}) \quad (1)$$

where $f(\underline{x})$, $q(\underline{x})$ and $h(\underline{x})$ are posynomials in the vector variable $\underline{x} = (x_1, \dots, x_m)$ and $a > 0$.

The above problem can be expressed as:

$$\text{minimize } g(\gamma) = f(x) + x_0^a h(x) \quad (2)$$

$$\text{subject to } x_0^{-1} q(x) \leq 1 \quad (3)$$

where x_0 is an additional independent variable and $\gamma = (x_0, x_1, \dots, x_m)$. It can be seen from the construction of $g(\gamma)$ and the constraint that $(x_1^1, x_2^1, \dots, x_m^1)$ minimizes $G(x)$ if and only if, $(q(x^1), x_1^1, \dots, x_m^1)$ minimizes $g(\gamma)$ subject to the given constraint. The constrained minimum value of $g(\gamma)$ is equal to the minimum value of $G(\underline{x})$. Thus the problem of minimizing $G(\underline{x})$ which is not necessarily a posynomial can be transformed to the form which permits the use of geometric programming.

Example 2.

Minimize the function

$$G(\underline{x}) = f(\underline{x}) + \frac{q(\underline{x})}{[v(\underline{x}) - h(\underline{x})]^a} \quad (4)$$

where f, q, h are posynomials, u is a posynomial with one term and $a > 0$.

The equivalent problem can be formed as

$$g(\gamma) = f(\underline{x}) + \frac{g(x)}{x_0^a} \quad (5)$$

subject to the constraint

$$\frac{x_0}{u(\underline{x})} + \frac{h(\underline{x})}{u(\underline{x})} \leq 1 \quad (6)$$

where x_0 is an additional independent variable and $\gamma = (x_0, x_1, \dots, x_m)$. Since $u(\underline{x})$ has only one term the form of the constraint permits use of geometric programming.

Example 3.

Minimize the function

$$G(\underline{x}) = f(\underline{x}) - u(\underline{x}) \quad (7)$$

where f and u are posynomials and u has one term. If it can be assumed that the minimum value of G is negative then the constraint.

$$x_0 + f(\underline{x}) - u(\underline{x}) \leq 0 \text{ is consistent} \quad (8)$$

It can be seen that x^1 minimizes $G(\underline{x})$ if and only if $\gamma^1 = [u(x^1) - f(x^1), x_1^1, \dots, x_m^1]$ maximizes the function

$$h(\gamma) = x_0 \quad (9)$$

subject to the constraint

$$x_0 + f(\underline{x}) - u(\underline{x}) \leq 0$$

This maximization problem is equivalent to the problem:

$$\text{Minimize } g(\tau) = \frac{1}{h(\tau)} = \frac{1}{x_0} \quad (10)$$

subject to the constraint

$$\frac{x_0}{u(\underline{x})} + \frac{f(\underline{x})}{u(\underline{x})} \leq 1 \quad (11)$$

Thus this reduces the problem to standard geometric programming form.

So far the examples showed the transformation which gives the exact solution of the problems. Following are some examples of approximate transformation which permits use of functions other than posynomials.

Example 4.

Any function $h(\underline{x})$ which is not a posynomial can be approximated to a single term posynomial. To do this it is necessary to make a rough estimate of the range of variability of each variable x_j . Let x_j^* be the geometric mean of this range. Then $(x_1^*, x_2^*, \dots, x_m^*)$ may be termed the operating point. Then $h(\underline{x})$ can be approximated as

$$h(\underline{x}) \approx h(\underline{x}^*) \left(\frac{x_m}{x_1^*}\right)^{A_1} \left(\frac{x_m}{x_2^*}\right)^{A_2}, \dots, \left(\frac{x_m}{x_m^*}\right)^{A_m} \quad (12)$$

where

$$A_j = \left(\frac{x_j}{h} \frac{\partial h}{\partial x_j}\right)_{\underline{x} = \underline{x}^*} \quad j = 1, \dots, m \quad (13)$$

This approximation is equivalent to expanding $\log h$ in a power series in terms of variables $Z_j = \log (x_j/x_j^*)$ and neglecting all but linear terms.

Example 5.

Approximation of $\log u$

The function $\log u$ is defined as:

$$\log u = \int_1^u \frac{1}{x} dx \quad (14)$$

On the other hand, if E is a positive number, then

$$\frac{u^E}{E} - \frac{1}{E} = \int_1^u \frac{x^E}{x} dx \quad (15)$$

On any interval between positive numbers, the function x^E is a uniform approximation to unity, provided E is sufficiently small. Hence, $\log u$ can be approximated by $E^{-1} (u^E - 1)$ for u in the same interval.

Example 6.

Approximation of an exponential function

Let the primal problem involve a function of the form

$$f(\underline{x}) = g(\underline{x}) + Ce^u(\underline{x}) \quad (16)$$

where g is a posynomial, C is a positive constant and $u(\underline{x})$ is a single term posynomial.

Using the well known relationship

$$e^u = \lim_{E \rightarrow \infty} \left(1 + \frac{u}{E}\right)^E \quad (17)$$

the function $f(\underline{x})$ can be written as

$$f_E(\underline{x}) = g(\underline{x}) + C \left(1 + \frac{u(\underline{x})}{E}\right)^E \quad (18)$$

where E is sufficiently large. This function is in the form of example 1. This can be reduced to standard geometric programming form by introducing a new variable $x_0 = 1 + u/E$. (19)

$$f(\underline{x}) \text{ can be replaced by the function } g(\underline{x}) + C x_0^E \quad (20)$$

and the additional constraint

$$x_0^{-1} + g^{-1} x_0^{-1} u(\underline{x}) \leq 1 \quad (21)$$

V. APPLICATIONS

In this section six different problems are considered to illustrate the procedure. The first is a simple hypothetical problem. The next four models are engineering design problems with different degrees of freedom and the last is a production scheduling model. The models are in posynomial form except for the last model, which is in the generalized posynomial form. Different optimization techniques are used to maximize the dual function as required by the nature of the problem. The results are compared and various computational difficulties are discussed. The last problem is solved by the Lagrangian algorithm.

A Simple Problem [17]

This simple problem can be stated as:

Minimize

$$y_0 = 1000 x_1 + 4 \times 10^9 x_1^{-1} x_2^{-1} \quad (1)$$

subject to

$$2.5 \times 10^5 x_2 + 9000 x_1^{-1} x_2^{-1} \leq 1 \quad (2)$$

$$x_1 > 0$$

$$x_2 > 0$$

The problem has 4 terms and 2 variables. Hence it has one degree of freedom. The dual function which is of the form of Eqn. (11) of Chapter II, can be written as:

$$V(\delta) = \left(\frac{C_1}{\delta_1}\right)^{\delta_1} \left(\frac{C_2}{\delta_2}\right)^{\delta_2} \left(\frac{C_3}{\delta_3}\right)^{\delta_3} \left(\frac{C_4}{\delta_4}\right)^{\delta_4} \times (\delta_3 + \delta_4)^{(\delta_3 + \delta_4)} \quad (3)$$

where $C_1 = 1000$

$$C_2 = 4 \times 10^5$$

$$C_3 = 2.5 \times 10^5$$

$$C_4 = 9000$$

The objective function has 2 terms. Hence the orthogonality condition is given by:

$$\delta_1 + \delta_2 = 1 \quad (4)$$

and the normality conditions are given by:

$$\delta_1 - \delta_2 - \delta_4 = 0 \quad (5)$$

$$\delta_2 + \delta_3 - \delta_4 = 0 \quad (6)$$

The minimum value of the function given by equation (1) is obtained by maximizing the dual function given by equation (3) subject to equality constraints (4), (5) and (6). These equality constraints are eliminated by expressing vector $\underline{\delta}$ as sum of a normality and a nullity vector as explained in Chapter II. Thus,

$$\underline{\delta} = \underline{b}^0 + r\underline{b}^1 \quad (7)$$

By the above substitution the problem is transformed to maximization with respect to single variable r and the positivity constraint

Table 1. Convergence rate of the sample problem by the Hooke and Jeeves method.

r_1	$V(\delta) \times 10^{12}$	No. of functional evaluation
.80	2.15	1
.90	4.57	5
.95	6.53	10
.97	7.73	16
.98	8.38	20
.99	8.70	26
1.00	8.99	30

$$\delta \geq 0$$

(8)

The problem was solved by using Hooke and Jeeves search procedure. The initial value chosen for r was 0.8. The accuracy, ϵ , as defined in Chapter II was chosen as .01. The convergence rate, i.e., the change of value of equation (3) with the number of functional evaluations is shown in Table 1.

The optimum value obtained was:

$$y_o^* = 8.99 \times 10^{12}$$

and the optimum primal variables were:

$$x_1 = 8.996 \times 10^9$$

$$x_2 = 2.0 \times 10^{-6}$$

The optimum primal variables were obtained using the following equations.

$$x_1 = \frac{y_o^* \delta_1}{c_1} \quad (9)$$

$$x_2 = \frac{\delta_3}{(\delta_3 + \delta_4)} \times c_3 \quad (10)$$

SEA POWER - HEAT EXCHANGER PROBLEM [6]

The conversion of the sun's radiant energy into useful power is a challenging field to many engineers. A stumbling block has been the extremely high capital cost of the equipment required to collect and concentrate this radiant energy.

Most of the solar energy is received by the upper layers

of the ocean. Energy from the sea could passibly be collected by a heat engine cycle which consists of direct evaporation of water from the upper layers, and, later, after passing through turbines, condensation on the cooler underlying water. The steam vapor pressure in equilibrium with the cool deep water is so low that extremely large turbines are required. An approach that circumvents the need for extremely large turbines is the use of an intermediate fluid, such as ammonia, which has a high vapor pressure at room temperature.

The avoidance of costly, large turbine is achieved only by introduction costly item, namely, the heat exchangers that allow heat to flow from the warm surface of the water to boiling ammonia and allow the same heat to flow from the condensing ammonia to cool deep water. Since the economic feasibility of this cycle depends primarily on the size of the required heat exchanger, our objective is to minimize the required surface area of heat exchangers for a sea power plant of a specific power capacity.

For derivation of the model the following nomenclature has been used:

A	=	area of heat exchangers
C	=	specific heat of water
f	=	friction coefficient
h	=	film coefficient of water
h'	=	film coefficient of ammonia
m	=	mass flow of water
P	=	power output heat engines

- P_N = net power output
 P_{Hx} = friction loss in heat exchangers
 P_{kE} = power loss by mass flow
 P_r = Prandtl number
 Q = heat extraction rate
 T = temperature of hot reservoir
 U = water flow rate
 $\alpha \Delta T$ = temperature gradient to heat engine
 $\beta \Delta T$ = total temperature drop across water boundary layers
 $\beta' \Delta T$ = total temperature drop across liquid ammonia
 $\gamma \Delta T$ = change in water temperature in heat exchangers
 ΔT = temperature difference in hot and cold reservoir
 ρ = density of water
 η = diffuser degradation
 ϵ = engine efficiency
 ϵ^1 = primemover's efficiency.

From thermodynamics the available power can be expressed as

$$P = \epsilon \epsilon \frac{\Delta T}{T} Q \quad (11)$$

The heat flux Q is restricted by the impedance of the water layer of essentially laminar flow that clings to the surface across which water is flowing. The heat characteristic of this film is specified as

$$Q = h A_{\frac{1}{2}} \beta \Delta T \quad (12)$$

The film coefficient h can be expressed as

$$h = \frac{f}{2 P_r^{2/3}} \rho C U \quad (13)$$

Thus an improvement of h can be obtained by increasing the velocity U but this is obtained only at the cost of an increase in power required to drive water through the heat exchangers. The power is expressed as

$$P_{Hx} = \frac{1}{2} f \rho U^3 (2A) \quad (14)$$

The heat flux Q is restricted also by the impedance of boiling ammonia; to boil ammonia at a finite rate, ammonia adjacent to the boiling heat exchanger must be slightly superheated. The relation between rate of boiling and the degree of superheat is given by

$$Q h^1 A^{\frac{1}{2}} \beta^1 \Delta T \quad (15)$$

The overall drop of water temperature in the heat exchangers is inversely proportional to the mass flow. We would like γ to approach zero so that α , β and β^1 could be larger. However, the smaller the value of γ is, the larger is the mass flow. Thus we must have

$$m C \gamma \Delta T = Q \quad (16)$$

and the loss of kinetic energy is

$$P_{KE} = 2 \gamma \left(\frac{1}{2} m U^2 \right) \quad (17)$$

Summarizing, the objective function to be minimized is the

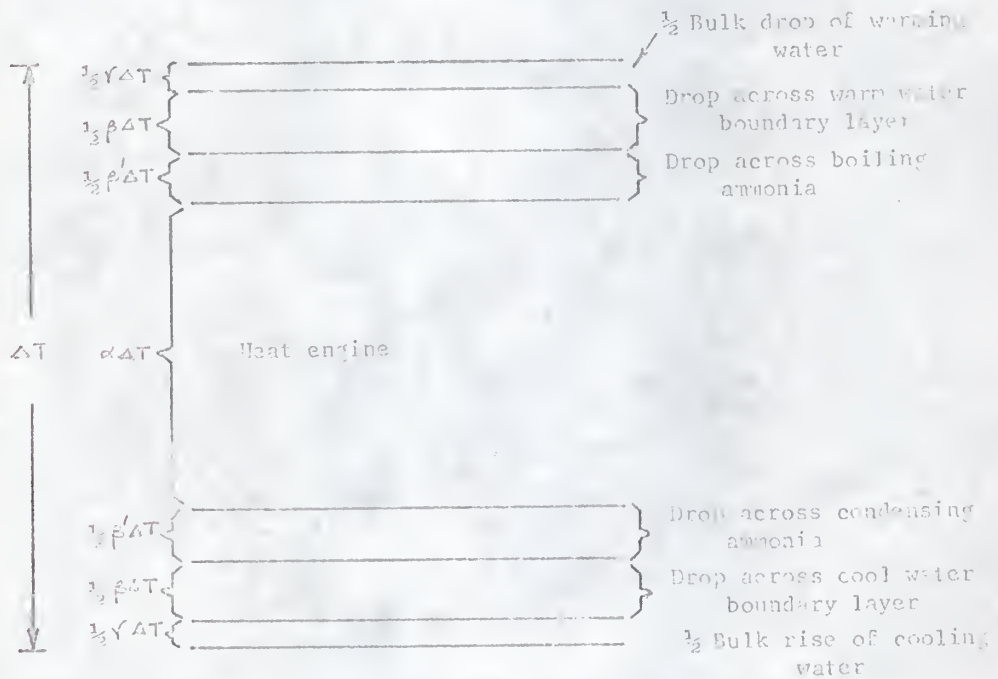


Fig. 1. Temperature distribution in the sea power problem [6].

area A, subject to constraints that A must be large enough to provide an adequate heat flux Q across the water boundary layers and across the liquid ammonia, and the heat flux Q must be large enough to provide not only P_N but also P_{Hx} and P_{KE} . Thus,

$$\frac{\epsilon \alpha \Delta T}{T} Q = P_N + \frac{1}{\epsilon} (P_{Hx} + P_{KE}) \quad (18)$$

The minimization of is to be made with respect to the variables Q, U, α , β , β^1 , γ subject to the preceding relationship as well as to the temperature distribution relationship.

$$\alpha + \beta + \beta^1 + \gamma = 1 \quad (19)$$

The temperature distribution is depicted in Fig. 1. The formulation of this problem is due to Zener and his associates and for detailed derivation the interested reader is referred to [6].

Solution by Geometric Programming

In geometric programming all constraints must be represented by inequalities rather than equalities. Although the constraints in the preceding section have been formulated as equalities, it is obvious they can be formulated as inequalities. Thus the heat flux across the thermal barriers must be equal to or greater than the heat flux Q through the heat engine, because some of the heat flux across the heat exchanger can bypass the heat engine if this permits a reduction in A.

Hence the constraint (12) can be formulated as

$$\frac{Q}{h A^{\frac{1}{2}} \beta \Delta T} \leq 1$$

and the constraint (15) as

$$\frac{Q}{h^{\frac{1}{2}} A^{\frac{1}{2}} \beta^{\frac{1}{2}} \Delta T} \leq 1$$

and the constraint (18) can be expressed as

$$\frac{P_N + (1/\epsilon^{\frac{1}{2}})(P_{HX} + P_{KE})}{(\epsilon \Delta T/T) Q} \leq 1$$

the = sign of constraint (19) can be changed into the \leq sign.

Hence the primal problem can be defined as

Minimize

$$C_1 A$$

subject to

$$C_2 \frac{Q}{UA \beta} \leq 1 \quad (20)$$

$$C_3 \frac{Q}{A \beta^{\frac{1}{2}}} \leq 1 \quad (21)$$

$$C_4 \frac{1}{Q\alpha} + C_5 \frac{AU^3}{Q\alpha} + C_6 \frac{U^2}{\alpha\gamma} \leq 1 \quad (22)$$

$$C_7 \alpha + C_8 \beta + C_9 \beta^{\frac{1}{2}} + C_{10} \gamma \leq 1 \quad (23)$$

where h has been eliminated from (20) and m from (22) by using (13) and (16) respectively. The constants $C_1, C_7, C_8, C_9, C_{10}$ are all unity and the constants C_2 through C_6 are

$$C_2 = \frac{4 P_r^{2/3}}{T^2 C \Delta T}$$

$$C_3 = \frac{2}{h^1 \Delta T}$$

$$C_4 = \frac{P_N}{\epsilon (\Delta T/T)}$$

$$C_5 = \frac{f \rho \times 10^7}{\epsilon \epsilon^1 (\Delta T/T)}$$

$$C_6 = \frac{\eta T \times 10^{-7}}{\epsilon \epsilon^1 C \Delta T^2}$$

The following numerical values are taken for the constants appropriate to water at room temperature:

$$P_r = 7$$

$$f = 1/125$$

$$\rho = 1 \text{ gm/cm}^3$$

$$C = 4.18 \text{ Joules/gm}$$

$$T = 300 \text{ }^\circ\text{K}$$

$$\epsilon = 0.6$$

$$\epsilon^1 = 0.6$$

$$\eta = 0.2$$

$$h^1 = 1.0 \text{ watt/cm}^2 \text{ }^\circ\text{C}$$

$$\Delta T = 11 \text{ }^\circ\text{C}$$

This gives the following values of the constants:

$$C_2 = 40$$

$$C_3 = 0.18$$

$$C_4 = 44.5 P_N$$

$$C_5 = 6.0 \times 10^{-8}$$

$$C_6 = 2.15 \times 10^{-8}$$

The above problem has 7 variables and a total of 10 terms. Hence it has 2 degrees of freedom. The dual problem to be maximized, $V(\delta)$, has the same form as Eqn. (11) which is subject to normality and orthogonality constraints having the same form as Eqn. (8) to Eqn. (10) of Chapter II. As the problem has 2 degrees of freedom the objective function of the dual problem can be expressed as a function of two independent variables r_1 and r_2 .

The problem has been solved by maximizing the dual function $V(\delta)$ by using Hooke and Jeeves search procedure. The convergence rate, i.e., the change in value of $V(\delta)$ with functional evaluation is shown in Table 3.

The problem has also been solved by differentiation. Two equations are obtained by differentiating $V(\delta)$ with respect to the independent variables and putting them equal to zero. Thus:

$$F_1 = \frac{dV(\delta)}{dr_1} = 0 \quad (24)$$

$$F_2 = \frac{dV(\delta)}{dr_2} = 0 \quad (25)$$

Equations (24) and (25) are solved by Newton-Raphson procedure. The convergence rate i.e., the change in the function values and change of the variables, are shown in Table 2 and plotted in Fig. 2 and 3.

The accuracy ϵ is chosen 0.001 for both Newton-Raphson and Hooke and Jeeves search procedure. The computer program is

Table 2. Convergence rate of the sea power problem by the Newton-Raphson method.

Iteration number	Variables		Function values	
	r_1	r_2	F_1	F_2
1	.64	.05	1.1	6.8
2	.46	.05	1.4	1.6
3	.42	.04	0.2	0.3
4	.42	.04	0.0	0.0

Table 3. Convergence rate of the sea power problem by the Hooke and Jeeves method.

Variables		$V(\delta)$	No. of functional evaluation
r_1	r_2		
.45	.02	124.69	0
.42	.07	124.79	7
.40	.04	126.03	33
.42	.04	126.72	41
.42	.04	126.75	59

Table 4. Optimum values of design parameters of the sea-power problem.

A	Q	U	α	β	β^1	γ
126.75	114.8	2.8	0.5	0.32	0.16	0.02

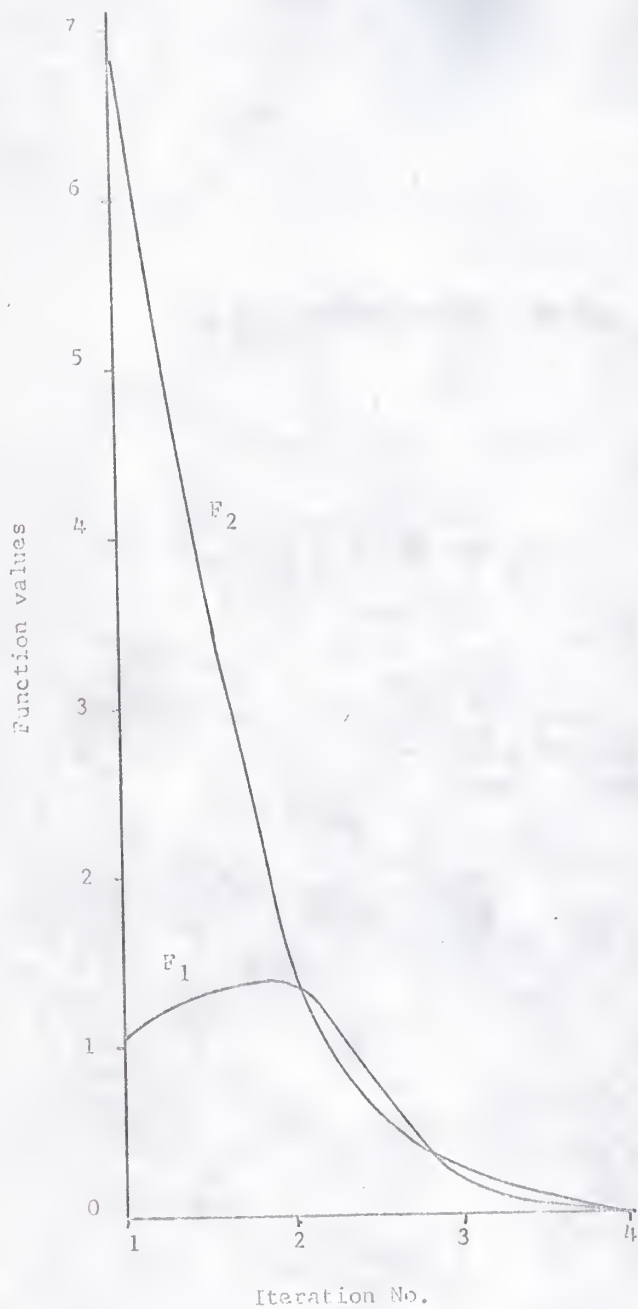


Fig. 2. The convergence rate of the sea power problem by the Newton-Raphson method.

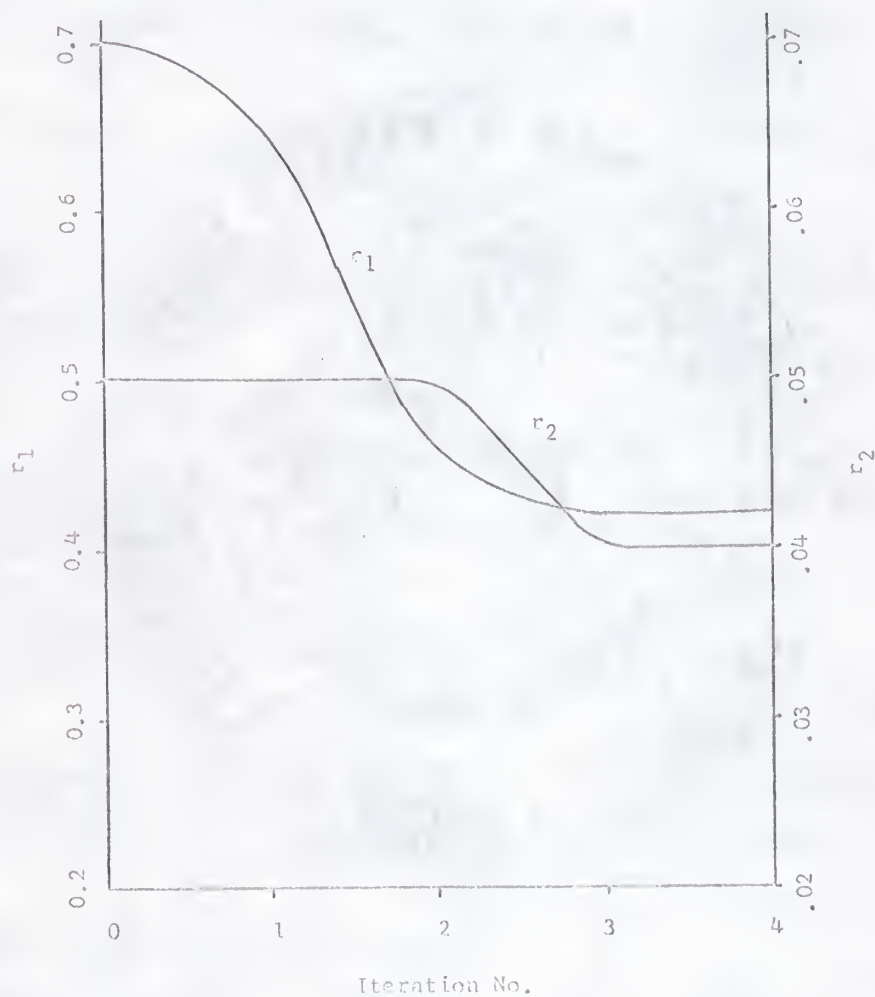


Fig. 3. The rate of change of variables in the sea power problem by the Newton-Raphsen method.

written in FORTRAN language and is given in the Appendix. The results are given in Table 4.

There is no significant difference in the results between the two methods. But the number of iterations required is much less in the Newton-Raphson procedure than in the search procedure. The computer time needed was 22 secs for the Hooke and Jeeves procedure and about 19 secs for the Newton-Raphson method. This is reasonable because the Newton-Raphson method requires more computation time. The Newton-Raphson method seems to be more efficient but, as will be seen later, it may not always converge.

CONDENSER DESIGN PROBLEM

This model is taken from a paper of Wilde and Avriel [1]. Detailed derivation of the formulation can be obtained from the original paper. In this model the design of a vapor condenser with fixed heat load is considered.

Consider a horizontal condenser in which a fluid having a given flow rate W is heated without phase change from temperature T_{b1} to T_{b2} by condensing saturated steam. Optimal design involves minimizing the annual cost of the condenser, consisting of three terms:

- 1) Cost of steam
- 2) Fixed charges on the condenser
- 3) Cost of pumping fluid through the condenser tube.

In the derivation of the model, the following nomenclature has been used:

A_i	=	inside heat transfer area
A_o	=	outside heat transfer area
B	=	pressure drop factor
C	=	annual cost
C_E	=	cost of electricity
C_F	=	fixed charges
C_H	=	unit cost of condenser surface
C_P	=	specific heat
C_{Pu}	=	pumping cost
C_S	=	cost of steam
D_i	=	inside tube diameter
D_o	=	outside tube diameter
f	=	fanning friction factor
g	=	specific gravity
k	=	thermal conductivity
l	=	tube wall thickness
L	=	tube length
N	=	No. of tubes in condenser
P_c	=	depreciation rate
P_F	=	plant factor
Q	=	condenser heat load
V	=	rate of heat transfer
R_F	=	fouling resistance
T_{bm}	=	mean bulk temperature
T_s	=	steam temperature
W	=	flow rate inside tubes
α	=	coefficient in steam cost equation

- α_1 = coefficient in steam cost equation
 ΔP = pressure drop
 ΔT_b = temperature rise in fluid in condenser
 ΔT_{mi} = mean drop through inside tube film
 ΔT_{mf} = mean drop through inside tube fouling
 ΔT_{mo} = mean drop through condensing film.

Assuming that the cost of steam can be expressed as a linear function of its saturation temperature, we can write:

$$C_s = \alpha_o Q + \alpha_1 T_s Q \quad (\$/\text{year}) \quad (26)$$

The fixed charges on the condenser are expressed as

$$C_F = C_H P_c A_o \quad (\$/\text{year}) \quad (27)$$

and the pumping cost is given by

$$C_{Pu} = \frac{C_E P W P_F}{P_N} \quad (\$/\text{year}) \quad (28)$$

The objective is to select the values of T_s , A_o and ΔP which minimize the total annual cost given by

$$C = C_s + C_F + C_{Pu} \quad (29)$$

The steam temperature can be written as

$$T_s = T_{bm} + T_{mo} + \Delta T_{mi} + \Delta T_{mf} \quad (30)$$

Without going into mathematical detail, from the theory of heat transfer, the first component, the cost of steam, can be expressed as

$$C_s = \alpha_o Q + \alpha_1 T_{bm} Q + \frac{\beta_1}{N^{7/6} D_o L^{4/3}} + \frac{\beta_2 D_i^{0.8}}{N^{0.2} L} + \frac{\beta_5}{N D_i L} \quad (31)$$

where

$$\beta_1 = \left(\frac{\alpha_1 P_F}{K_f} \right) (W C_P \Delta T_b)^{7/3} \left(\frac{\mu_f}{2 \lambda_f^2 g} \right)^{1/3} \left(\frac{1}{0.725} \right)^{4/3} \quad (32)$$

and

$$\beta_2 = \frac{(\alpha_1 P_F \Delta T_b)^2 (\mu)^{0.4} (C_P)^{1.5} W^{1.2}}{4^{0.8} (0.023) k^{0.6} \pi^{0.2}} \quad (33)$$

and

$$\beta_5 = \frac{\alpha_1 Q q R_F}{\pi} \quad (34)$$

The fixed charges can be expressed as

$$C_F = \beta_3 N D_o L \quad (35)$$

where

$$\beta_3 = \pi C_H P_c \quad (36)$$

and the pumping cost can be expressed as

$$C_{Pu} = \frac{\beta_4 L}{D_i^{1.8} N^{1.8}} \quad (37)$$

where

$$\beta_4 = \frac{32 \times 0.046 C_E B P_F W^{2.8} (\mu/4)^{0.2}}{g \eta \rho^2 (\pi)^{1.8}} \quad (38)$$

Total annual cost is given by

$$C = \alpha_o Q + \alpha_1 T_{bm} Q + \frac{\beta_1}{N^{7/6} D_o L^{4/3}} + \frac{\beta_2 D_i^{0.8}}{N^{0.2} L} + \beta_3 N D_o L + \frac{\beta_4 L}{D_i^{1.8} N^{1.8}} + \frac{\beta_5}{N D_i L} \quad (39)$$

Since the first two terms are constants only the last five terms may vary and are subject to optimization. Thus the variable cost function,

$$C^{\#} = \frac{\beta_1}{N^{7/6} D_o L^{4/3}} + \frac{\beta_2 D_i^{0.8}}{N^{0.2} L} + \beta_3 N D_o L + \frac{\beta_4 L}{D_i^{4.8} N^{1.8}} + \frac{\beta_5}{N D_i L} \quad (40)$$

From practical consideration the constraint on inside and outside diameter of the tubes is:

$$D_o - D_i \geq 2l \quad (41)$$

This can be written in the form of a geometric programming constraint as

$$\frac{\beta_6}{D_o} + \frac{\beta_7 D_i}{D_o} \leq 1 \quad (42)$$

where $\beta_6 = 2l$ and $\beta_7 = 1$

Also from a practical standpoint it was found D_o can not exceed 1 inch. Hence this constraint can be included as:

$$D_o \leq D_o \text{ max}$$

$$\text{or } \beta_8 D_o \leq 1 \quad (43)$$

where $\beta_8 = 1/D_o \text{ max} = 12$

The following numerical values have been taken

$$W = 500,000 \text{ lbs/hr.}$$

$$T_{bl} = 195^\circ \text{F}$$

$$\begin{aligned}
T_{b2} &= 205^{\circ}\text{F} \\
T_{bm} &= 200^{\circ}\text{F} \\
\alpha_1 &= 10^{-9} \text{ \$/BTU} \cdot ^{\circ}\text{F} \\
C_H &= 5 \text{ \$/sq. ft.} \\
C_E &= 10^{-2} \text{ \$/kw-hr.} \\
P_c &= 0.1 \\
P_f &= 7884 \text{ hr/yr} \\
\eta &= 0.8 \\
\rho &= 60.13 \text{ lb/cu.ft.} \\
\mu &= 0.20 \text{ cp.} \\
k &= 0.393 \text{ BTU/hr-sq.ft.} \cdot ^{\circ}\text{F/ft.} \\
C_p &= 1.01 \text{ BTU/lb} \cdot ^{\circ}\text{F} \\
T_f &= 210^{\circ}\text{F} \\
\rho_f &= 59.88 \text{ lb/cu.ft.} \\
\mu_f &= 0.26 \text{ cp.} \\
k_f &= .393 \text{ BTU/hr-sq.ft.} \cdot ^{\circ}\text{F/ft.} \\
\lambda &= 960 \text{ BTU/lb.} \\
l &= .049 \text{ inch} \\
B &= 1.2 \\
R_f &= 5.68 \times 10^{-4}
\end{aligned}$$

The following values of the constants are obtained:

$$\begin{aligned}
\beta_1 &= 172,400 \\
\beta_2 &= 97,790 \\
\beta_3 &= 1.57 \\
\beta_4 &= 0.0382 \\
\beta_5 &= 38380
\end{aligned}$$

$$\beta_6 = 0.00817$$

$$\beta_7 = 1.0$$

$$\beta_8 = 12.0$$

Solution by geometric programming

The problem consists of a total of 8 terms and 4 variables. Hence it has 3 degrees of freedom. The dual problem to be maximized has the same form as Eqn. (11) which is subject to normality and orthogonality constraints having the form of Eqn. (8) to Eqn. (10) of Chapter II. As the problem has 3 degrees of freedom the dual objective function can be expressed as a function of 3 independent variables r_1, r_2, r_3 . Other variables are eliminated by the use of linear equality constraints.

The problem has been solved by maximizing the dual function $V(\delta)$ by using Hooke and Jeeves search procedure. The convergence rate, i.e., the change of value of $V(\delta)$ with the number of functional evaluations during the search is shown in Table 5 and is plotted in Fig. 4.

The problem did not converge to an optimal with the Newton-Raphson method. The initial value $r_i = 1 \quad i = 1, \dots, 3$ was tried. The difficulty with the Newton-Raphson method was in the first step where the values of r_i violated the positivity constraints

$$b^0 + \sum_{i=1}^3 r_i b^i \geq 0$$

so widely that they could not be corrected.

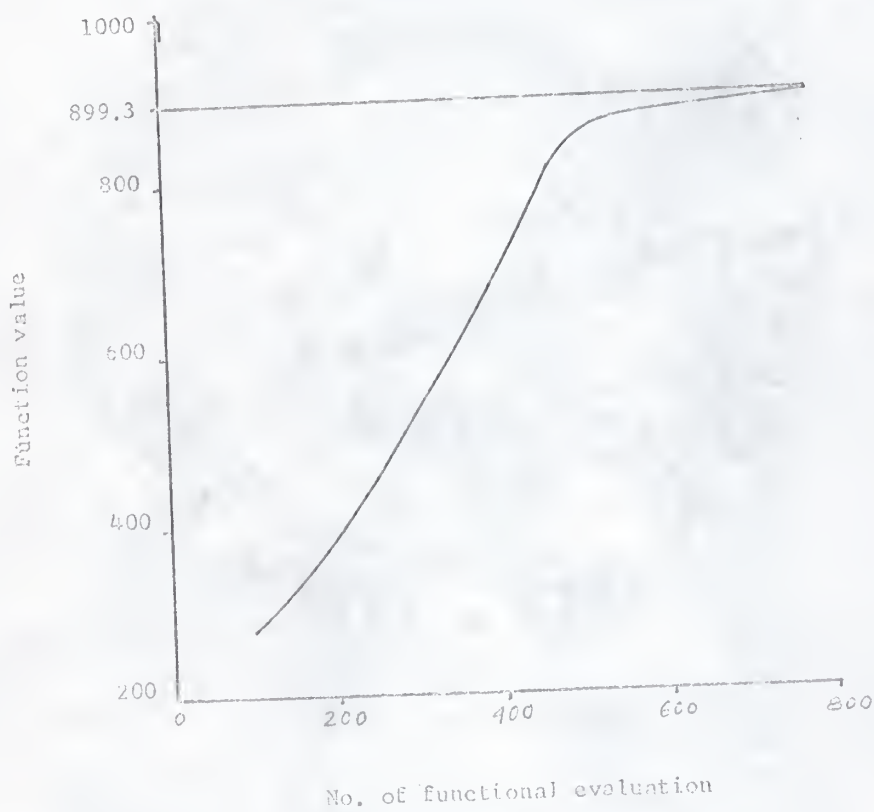


Fig. 4. The convergence rate of the condenser problem by the Hooke and Jeeves method.

Table 5. Convergence rate of the condenser design problem by the Hooke and Jeeves method.

Variables			Function Value	No. of Functional Evaluation
r_1	r_2	r_3		
.810	.210	.790	196.64	5
.800	.370	.775	272.83	99
.620	.365	.595	384.53	200
.440	.355	.415	532.16	304
.360	.355	.335	608.99	352
.270	.345	.245	702.24	403
.190	.340	.165	786.06	449
.100	.330	.075	867.08	501
.085	.325	.060	879.67	555
.083	.326	.057	880.69	602
.067	.325	.041	890.07	650
.045	.322	0.18	898.26	701
.035	.320	.008	899.33	783

The following results were obtained:

Minimum variable annual cost = 899.33 \$/year. The optimal design parameters were:

$$D_0 = 1 \text{ inch}$$

$$D_1 = .90 \text{ inch}$$

$$N = 114.16$$

$$L = 27.48 \text{ feet}$$

The optimal dual variables were:

δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8
.1095	.1934	.4564	.0617	.1790	.0351	.3203	.0085

The accuracy, ϵ , for Hooke and Jeeves search was chosen 0.001. The computation time taken was 115 secs.

CHEMICAL EQUILIBRIUM PROBLEM

This model is taken from a paper of Passy and Wilde [13]. According to the minimum free energy principle, a chemical system is in equilibrium at constant pressure and temperature if and only if its free energy is a minimum. To formulate this problem mathematically, let a chemical system have P phases, G_1, G_2, \dots, G_p and let the chemical species occurring in phase G_k be $A_{k1}, A_{k2}, \dots, A_{k,I(k)}$ where $I(k)$ is the number of species in G_k . Let n_{ki} be the number of moles of species A_{ki} in phase G_k . Then define a row vector N_k representing the composition of phase G_k whose components are $n_{k1}, n_{k2}, \dots, n_{I(k)}$. From these construct n :

$$n = [N_1, N_2, \dots, N_p]$$

In this notation the mass balance equation for each chemical element B_j is given by

$$\sum_{m=1}^P \sum_{k=1}^{I(m)} a_{mkj} n_{mk} = b_j \quad j = 1, 2, \dots, r \quad (44)$$

where a_{mkj} is the number of atoms of element B_j in chemical species A_{mk} , b_j is the mass in gram atoms of chemical element B_j , and r is the number of different chemical elements in the system.

The Gibbs free energy $G(n)$ of the system is given by

$$G(n) = RT \sum_{k=1}^P \sum_{i=1}^{I(k)} n_{ki} \left(\ln \frac{n_{ki}}{N_k} - \ln C_{ki} \right) \quad (45)$$

$$\text{where } N_k = \sum_{i=1}^{I(k)} n_{ki} \quad \text{for } k = 1, \dots, P$$

and the second term is the standard free energy. The equilibrium concentration n^* is found by minimizing $G(n)$ subject to (32) and the natural constraints,

$$n_{ki} \geq 0 \quad \text{for all } i \text{ and } k$$

This minimizing problem, which has a unique solution in the trivial case, is equivalent to the dual geometric programming problem since

$$V(n_{ol}, n) = \exp\left(\frac{-G(n_{ol}, n)}{RT}\right) = \exp\left(\frac{-G(n)}{RT}\right) \quad (46)$$

where (n_{ol}, n) is the vector generated by augmenting one

component n_{ol} to the vector n ; the corresponding coefficient C_{ol} is set equal to unity.

The mass balance equation can be written as

$$n_{ol} = 1 \quad (47)$$

$$\sum_{m=0}^p \sum_{k=1}^{I(m)} A_{mkj} n_{mk} = 0 \quad j = 1, \dots, r \quad (48)$$

where $a_{oij} = -b_j$ and $I(0) = 1$

These are identified as the normality and orgonality conditions as equations (9) and (10) of Chapter II. Hence the primal objective function of a chemical equilibrium problem can be identified as

$$\sum_{j=1}^r t_j \quad (49)$$

subject to constraints

$$h_m(t) \leq 1 \quad m = 1, \dots, P \quad (50)$$

$$\text{where } h_m(t) = \sum_{k=1}^{I(m)} C_{mk} \sum_{j=1}^r t_j a_{mkj} \quad (51)$$

and r is the number of different chemical elements B_j , and b_j is the mass of chemical element B_j . t_j is the corresponding primal variable.

The problem discussed here is due to White, Johnson and Danzing [16]. The stoichiometric mixture of hydrazine and oxygen at 3500 °k and a pressure of 750 psi is considered. The initial amounts of hydrogen, oxygen and nitrogen are 2, 1 and 1

respectively.

So the objective function is

$$g_0(t) = t_1^{-2} \times t_2^{-1} \times t_3^{-1} \quad (52)$$

$$g_1(t) = C_1 t_1 + C_2 t_1^2 + C_3 t_1^2 t_2 + C_4 t_3 + C_5 t_3^2 + C_6 t_3 t_1 \\ + C_7 t_3 t_2 + C_8 t_2 + C_9 t_2^2 + C_{10} t_1 t_2 \leq 1 \quad (53)$$

The various possible constituents at equilibrium and the corresponding C_s are given as

$$H = 4.411 \times 10^2$$

$$H_2 = 2.846 \times 10^7$$

$$H_2O = 6.160 \times 10^{14}$$

$$N = 3.703 \times 10^2$$

$$N_2 = 7.107 \times 10^{10}$$

$$NH = 3.225 \times 10^6$$

$$NO = 2.930 \times 10^6$$

$$O = 4.471 \times 10^4$$

$$O_2 = 3.796 \times 10^{11}$$

$$OH = 4.289 \times 10^9$$

$$\text{putting } y_1 = 10^3 t_1 \quad y_2 = 10^6 t_2 \quad y_3 = 10^5 t_3$$

The objective is to minimize

$$g_0(y) = 10^{16} y_1^{-2} y_2^{-1} y_3^{-1} \quad (54)$$

subject to

$$\begin{aligned}
g_1(y) = & .4411 y_1 + 28.46 y_1^2 + 616 y_1^2 y_2 + 0.03703 y_3 \\
& + 710.7 y_3^2 + .3225 y_1 y_3 + 2.93 y_2 y_3 + .04471 y_2 \\
& + 0.3796 y_2^2 + 4.289 y_1 y_2 \leq 1
\end{aligned} \tag{55}$$

Solution by geometric programming

The problem has 11 terms and 3 variables, and hence a degree of freedom 7. The problem is solved by geometric programming using Hooke and Jeeves search procedure. As the degree of freedom is 7, the dual problem can be expressed as a function of 7 independent variables, r_i , $i = 1, \dots, 7$. The convergence rate of Hooke and Jeeves is shown in Table 6 and the same data is plotted in Fig. 5.

The Newton-Raphson method did not converge for this problem. An initial value of $r_i = 1$, $i = 1, \dots, 7$ was tried. The difficulty with the Newton-Raphson method occurred after the first step when the value of the variables r_i violated the positivity constraints $\underline{b}^0 + \sum_{i=1}^7 r_i \underline{b}^i \leq 0$

so widely that they could not be corrected.

The computation time for Hooke and Jeeves procedure was 239 secs. The accuracy ϵ was chosen as 0.001.

The following values of primal variables were obtained at the optimum.

y_1	y_2	y_3
0.056229	0.245371	0.025817

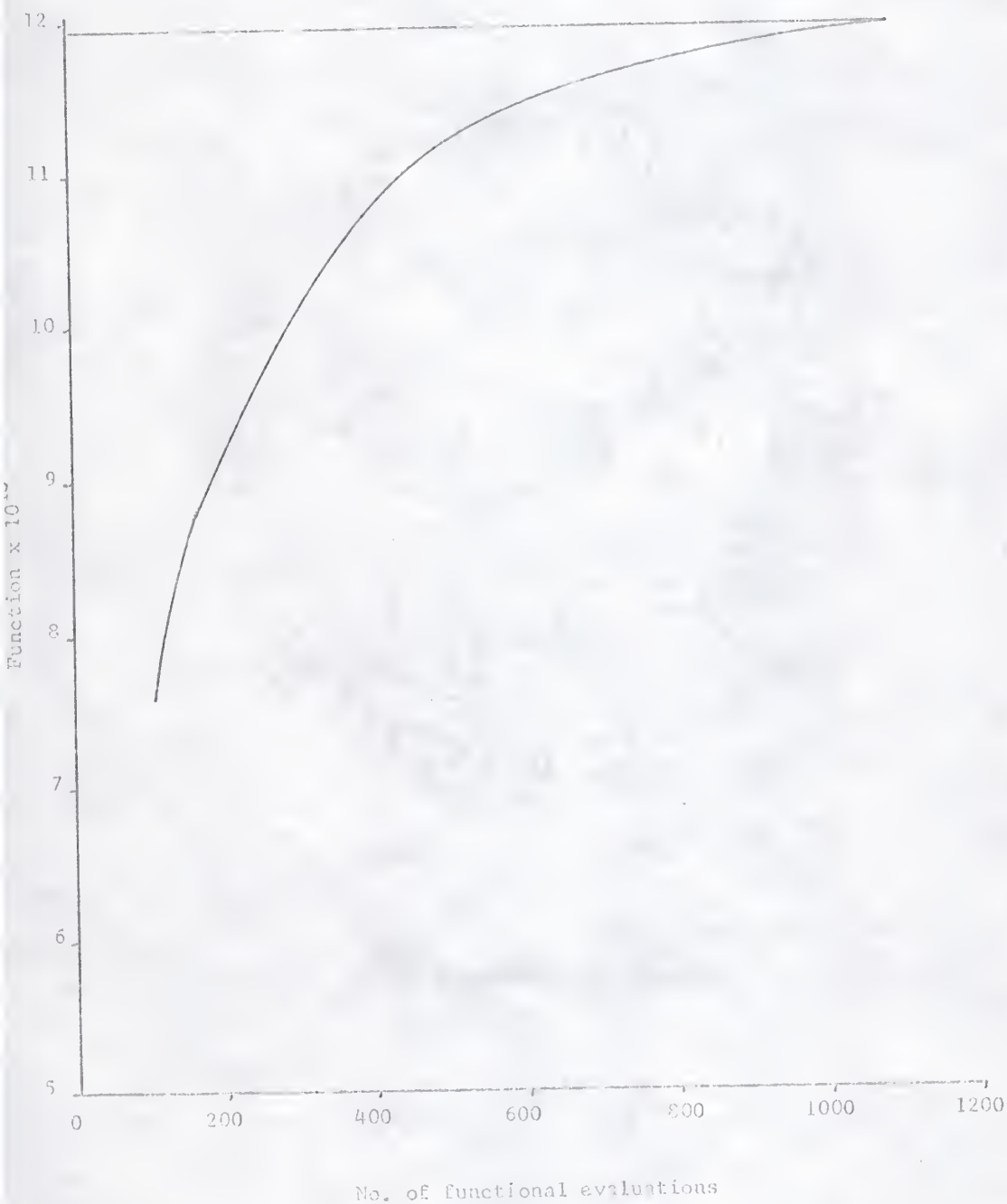


Fig. 5. The convergence rate of the chemical equilibrium problem by the Hooke and Jeeves method.

Table 6. Convergence rate of the chemical equilibrium problem by the Hooke and Jeeves method.

Variables							Function Value x 10 ¹³	No. of functional evaluation
r ₁	r ₂	r ₃	r ₄	r ₅	r ₆	r ₇		
.150	.250	.150	.050	.100	.100	.150	1.08	14
.050	.400	.050	.100	.050	.050	.050	7.61	112
.050	.412	.012	.150	.012	.012	.075	9.49	224
.044	.431	.006	.125	.012	.019	.081	10.28	318
.050	.447	.003	.097	.016	.022	.084	10.89	404
.044	.459	.003	.075	.016	.028	.087	11.30	522
.044	.461	.002	.073	.016	.028	.089	11.35	606
.056	.473	.002	.048	.017	.034	.094	11.62	701
.050	.478	.002	.041	.017	.036	.095	11.71	804
.044	.483	.001	.032	.018	.037	.096	11.77	1001
.041	.486	.001	.026	.018	.037	.098	11.79	1091

The optimum value of objective function was 11.79×10^{13} .

The optimal dual variables were

$$\begin{aligned}\delta_1 &= 1.00 \\ \delta_2 &= .002 \\ \delta_3 &= .146 \\ \delta_4 &= .784 \\ \delta_5 &= .041 \\ \delta_6 &= .486 \\ \delta_7 &= .001 \\ \delta_8 &= .026 \\ \delta_9 &= .018 \\ \delta_{10} &= .037 \\ \delta_{11} &= .097\end{aligned}$$

The dual variables δ_2 to δ_{11} represent the optimum equilibrium concentrations in moles of the corresponding species.

TRANSFORMER PROBLEM

This model is taken from a Westinghouse Research Report [7]. The explanation of the model and identification of variables are not disclosed. The problem is stated as follows:

Minimize

$$\begin{aligned}U(t) &= .2007 t_3 t_4 t_5 + .2597 t_1 t_2 t_6 \\ &+ 3.69 \times 10^9 t_6 / t_1 t_2 t_3^2 t_4^2\end{aligned}\quad (56)$$

subject to

$$g_1(t) = 4 t_1 / t_5 + 6 t_2 / t_5 + 4 t_3 / t_5 \leq 1 \quad (57)$$

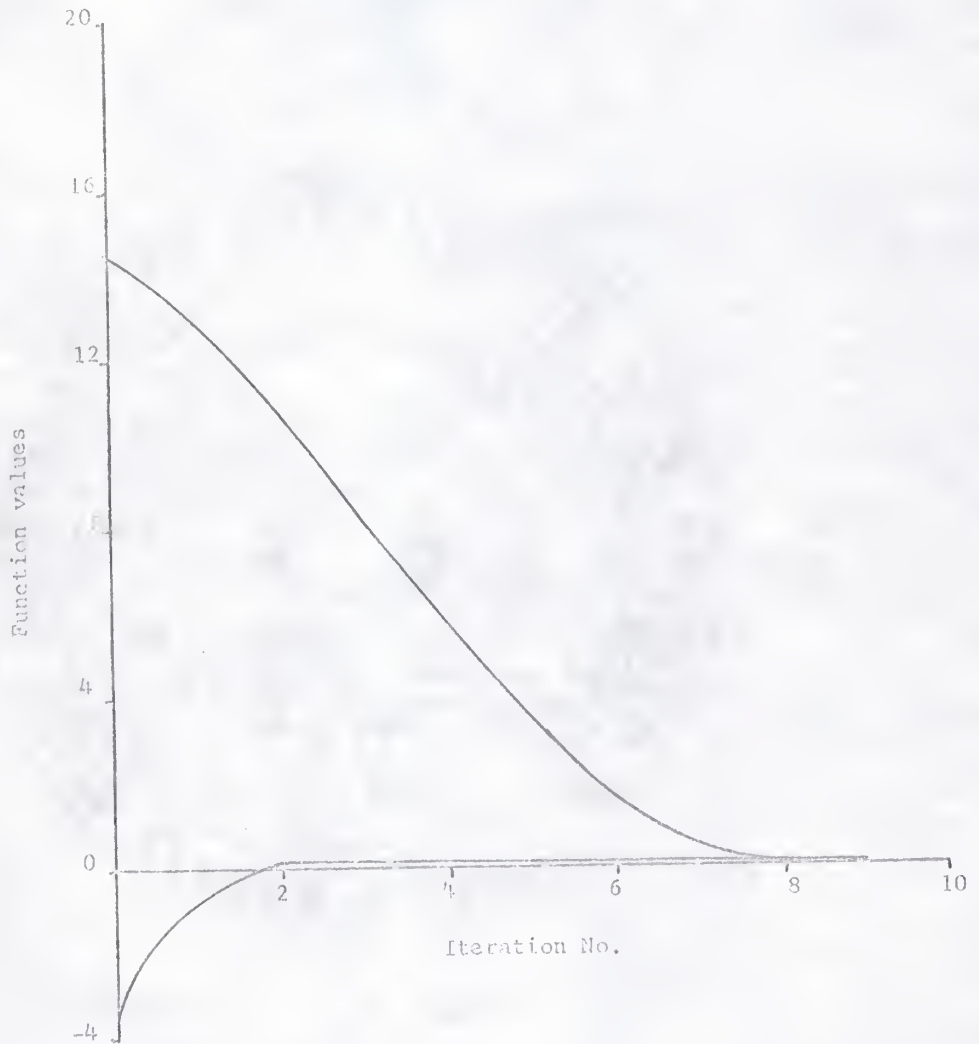


Fig. 6. The convergence rate of the transformer problem by the Newton-Raphson method.

Table 7. Convergence rate of the
Transformer problem by the
Hooke and Jeeves method.

Variables		$V(\delta)$	No. of Functional Evaluation
r_1	r_2		
.200	.200	55955.00	1
.400	.100	58391.38	10
.325	.075	65932.50	20
.337	.100	66420.06	30
.312	.106	66671.75	40
.322	.100	66694.62	50
.317	.103	66698.93	60
.320	.103	66703.75	70

Table 8. Convergence rate of the Transformer problem by the Newton-Raphson method.

Iteration	Variables		Function Values	
	r_1	r_2	F_1	F_2
0	.1500	.2500	-3.5099	14.5100
1	.2051	.2500	-.8765	12.8205
2	.2243	.2500	.0992	10.8018
3	.2228	.2497	.0021	8.2646
4	.2243	.2469	.0002	5.9995
5	.2372	.2294	.0065	3.8335
6	.2793	.1691	.0547	1.6322
7	.3173	.1091	.0492	.1677
8	.3193	.1033	.0010	.0004
9	.3193	.1033	0.000	0.000

Table 8a. Computational aspects of problems 1 to 5.

Problem No.	Total No. of terms	No. of variables	Degree of difficulty	Procedure for maximizing dual function			
				Newton - Raphson	Hooke and Jeeves		
				Computer time (sec.)	No. of iterations	Computer time (sec.)	No. of functional evaluation
1	4	2	1	-	-	2.5	30
2	10	7	2	19	4	22	59
3	8	4	3	-	-	115	753
4	11	3	7	-	-	239	1091
5	9	6	2	16	9	13	70

$$g_2(t) = 6 t_3/t_6 + 6 t_4/t_6 = 9.424 t_1/t_6 \leq 1 \quad (58)$$

Solution by geometric programming

The above problem has 9 terms and 6 variables, and hence there are 2 degrees of freedom. The dual function can be expressed with two independent variables r_1 and r_2 . The problem has been solved by maximizing the dual function by Hooke and Jeeves procedure. The convergence rate for the search is shown in Table 7. The problem has also been solved by the Newton-Raphson procedure and the convergence rate is shown in Table 8 in which the functions F_1 and F_2 are derivatives of the dual function with respect to r_1 and r_2 respectively. The same data are plotted in Fig. 6. The accuracy is chosen as 0.001 in both the Newton-Raphson and the Hooke and Jeeves procedure.

The following results are obtained:

Minimum value of the objective function is 66703.93

Optimum primal variables are:

t_1	t_2	t_3	t_4	t_5	t_6
19.074	29.690	42.524	6.601	426.532	70.625

And the optimum dual variables are:

δ_1	δ_2	δ_3	δ_4	δ_5	δ_6	δ_7	δ_8	δ_9
.43	.19	.38	.08	.18	.17	.15	.32	.10

PRODUCTION - INVENTORY PROBLEM

This is a hypothetical model in which the optimum policy regarding production and inventory levels are to be determined

Let

$$x(t) = I(t) \quad (62)$$

$$z(t) = P(t) \quad (63)$$

Then Equation (59) becomes

$$dx(t)/dt = z(t) - Q(t)$$

From this the difference equation can be written as

$$x(t + \Delta t) = x(t) + (z(t) - Q(t)) \Delta t \quad (64)$$

Dividing the entire time period into five stages, the integral equation for cost (50) can be written as

$$C_T = \sum_{i=1}^5 [C_I (I_m - x_i)^2 + C_P (P_m - z_i)^2] \Delta t \quad (65)$$

and difference Eqn. (64) can be written as

$$x_i = x_{i-1} + (z_i - Q_i) \Delta t \quad i = 1, \dots, 5 \quad (66)$$

The numerical values assumed are

$a = 2$	$b = 1$	$c = 5$
$C_I = 0.1$	$I_m = 10$	$P_m = 5$
$C_P = .01$	$t_o = 0$	$t_f = 1$
$t = .2$		

To apply geometric programming the objective function has to be expressed in polynomial form. Thus rewriting Eqn. (65)

$$C_T = \sum_{i=1}^5 C_I \Delta t x_i^2 + \sum_{i=1}^5 C_P \Delta t z_i^2 - \sum_{i=1}^5 2 C_I I_m \Delta t x_i - \sum_{i=1}^5 2 C_P P_m \Delta t z_i + \text{Constant} \quad (67)$$

where $\text{Constant} = C_I I_m^2 + C_P P_m^2$

The problem is to minimize the variable portion of Eqn. (67) subject to the constraints (66). An attempt was made to solve the problem by geometric programming. The problem has 34 terms and 10 variables, and hence the degree of freedom is 23. The problem is not in posynomial form; hence the extension of geometric programming as discussed in Chapter II had to be used. The attempt was unsuccessful because of the difficulty in obtaining an initial feasible solution r_j . As the problem has 23 degrees of freedom. A feasible solution of r_j , $j = 1, 23$ has to be found which satisfies the inequality constraints given by Equations (21) and (24) of Chapter II, which are

$$\infty > \delta_{kt} \geq 0 \quad (68)$$

$$\text{and } \delta_{ko} = \sigma_k \sum_{t=1}^{T_k} \sigma_{kt} \delta_{kt} \geq 0 \quad k = 1, \dots, p. \quad (69)$$

where

$$\underline{\delta} = \underline{b}^0 + \sum_{j=1}^d r_j \underline{b}^j$$

$$\sigma_k = \pm 1$$

$$\sigma_{kt} = \pm 1$$

and d is degree of freedom.

The problem has 34 terms and 5 constraints. So an initial solution of \underline{r} has to satisfy 39 inequality constraints. The Hooke and Jeeves search was used to maximize each δ_{ko} given by Equation (69) subject to constraints (68) until $\delta_{ko} \geq 0$. Hooke

and Jeeves search is not efficient in handling a large number of inequality constraints. Due to this difficulty an initial feasible solution could not be obtained by this method.

The above problem was solved successfully by Lagrangian polynomial optimization technique [3]. To use this technique, the problem has to be expressed in the following form

$$\text{Minimize } g_0 \quad (70)$$

$$\text{subject to } g_m = 1 \quad m = 1, \dots, M \quad (71)$$

$$\text{where } g_m = \sum_{t=1}^T \sigma_{mt} C_{mt} \prod_{n=1}^N X_n^{\Lambda_{mtn}} \quad m = 0, 1, \dots, M \quad (72)$$

$$\text{where } C_{mt} > 0$$

$$\Lambda_{mtn} \text{ is any real number}$$

$$\sigma_{mt} = \pm 1$$

The objective function of the above problem given by Equation (67) is in the form of (70). The constraints as given by Eqn. (66) can be transformed according to the requirement for this technique as follows:
rewriting Equation (66) as

$$\frac{x_1}{x_0 + Q_1 t} - \frac{z_1 t}{x_0 + Q_1 t} = 1 \quad (73)$$

and

$$\frac{x_{i-1}}{Q_1 t} - \frac{x_i}{Q_1 t} + \frac{z_i}{Q_1 t} = 1 \quad i = 2, \dots, 5 \quad (74)$$

constraints (73) and (74) are in the required form and the algorithm as described in Chapter III can directly be applied.

A difficulty in this problem, was that the function overshoot the minimum for some starting values because of too large a step size. This difficulty was overcome by using a forcing procedure which restricts the step size to a certain maximum of the variables. Various limits were tried on different starting values. The number of iterations required to converge to the optimum for each of these cases is given in Table 11. The convergence rates for two typical starting values are given in Tables 9 and 10. The optimum values of productions and inventories are given in Table 12, and are plotted in Figs. 7 and 8. The function is assumed to have converged when all components of the error vector as defined in Chapter III are less than or equal to 0.0001.

The optimum total cost obtained is

$$C_T = .374$$

Table 9. Typical convergence rate of the inventory problem with starting values $x_i = 10$, $i=1, \dots, 10$.

Iteration No.	Cost Percentage forcing			
	No forcing	20	50	100
0	-9.9999	-9.9999	-9.9999	-9.9999
1	-9.8864	-9.9290	-9.8864	-9.8864
2	-9.8738	-9.616	-9.8738	-9.8738
3	-9.8759	-9.8750	-9.8759	-9.8759
4	-9.8759	-9.8759	-9.8759	-9.8759
5	---	-9.8759	-9.8759	-9.8759

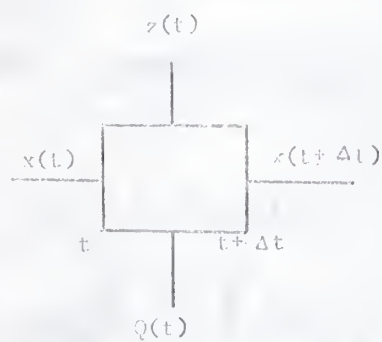
Table 10. Typical convergence rate for the inventory problem with starting values
 $x_i = 5.0 \quad i = 1, \dots, 10.$

Iteration No.	Cost			
	Percentage forcing			
	No forcing	20	50	100
0	-7.7499	-7.7499	-7.7499	-7.7499
1	703.2946	-8.6153	-9.2914	-9.5404
2	1022.4440	-9.3822	-7.4436	-9.6252
3	378.3039	-9.0118	-8.2156	-9.8360
4	93.0727	-9.5835	-6.3799	-9.8680
5	3.3208	-9.6459	-7.7937	-9.8754
6	-7.2251	-9.0581	-7.4099	-9.8758
7	-9.0035	-8.9299	-7.4086	-9.8758
8	-8.2692	-9.3047	-7.3310	
9	-9.5499	-9.3213	-7.3615	
10	-9.7332	-9.2770	-7.3502	
11	-9.7480	-9.1200	-7.3435	
12	-9.7644	-9.1454	-7.3397	
13	-9.7679	-9.1501	-7.3374	
14	-9.7678	-9.1504	-7.3360	
15	-9.7677	-9.1506	-7.3353	
16	-9.7677	-9.1503	-7.3348	

Table 11. Effect of forcing on convergence.

Starting Value	No. of Iterations				
	Percentage forcing				
	No forcing	10	20	50	100
	No. Conv.	--	No. Conv.	No. Conv.	No. Conv.
4.0	10	--	No. Conv.	--	--
4.4	9	--	--	--	--
5.0	No. Conv.	20	No. Conv.	No. Conv.	8
6.0	6	--	--	--	--
6.5	5	--	--	--	--
8	No. Conv.	--	7	7	No. Conv.
9	No. Conv.	--	7	11	No. Conv.
10	4	--	5	5	5

Table 12. Optimum inventory and production levels.



$t - t + t$	$x(t)$	$z(t)$	$q(t)$
0.0 - 0.2	5.0	14.01	2.0
.2 - .4	7.38	8.77	2.30
.4 - .6	8.63	4.14	2.50
.6 - .8	9.40	3.93	2.70
.8 - 1.	9.84	4.62	2.90
1. -	10.20	--	

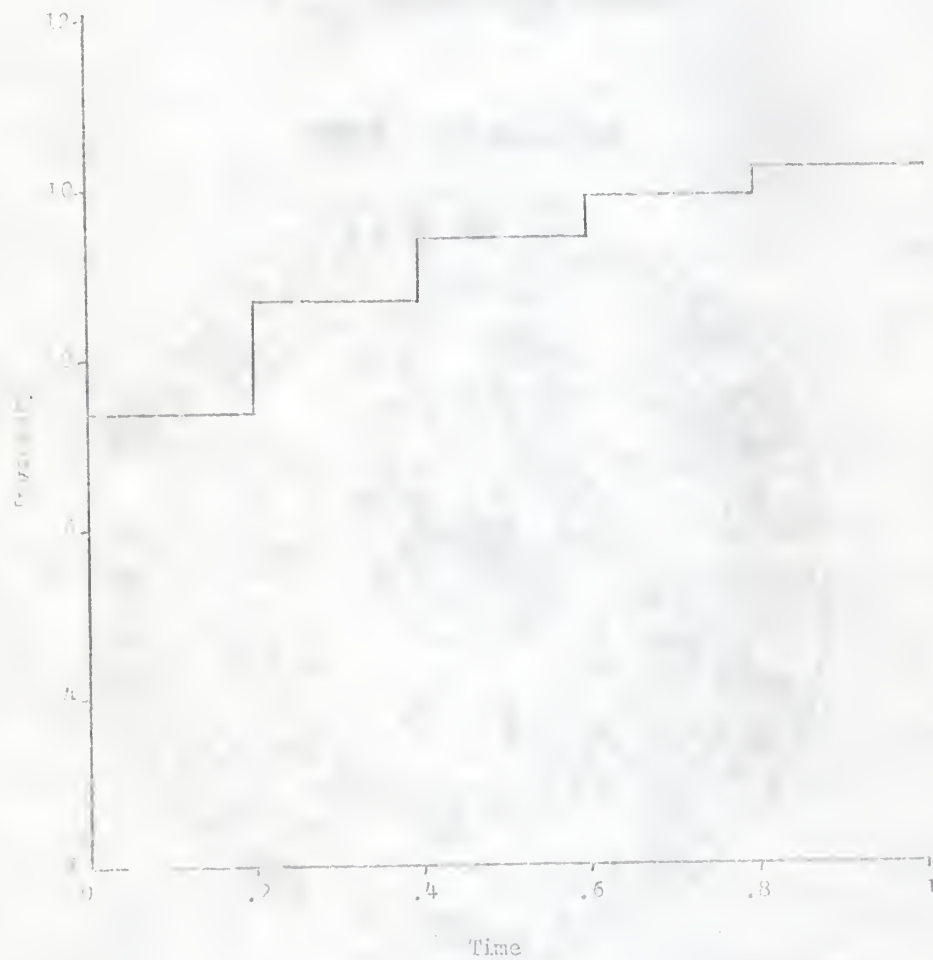


Fig. 7. Optimal inventory levels.

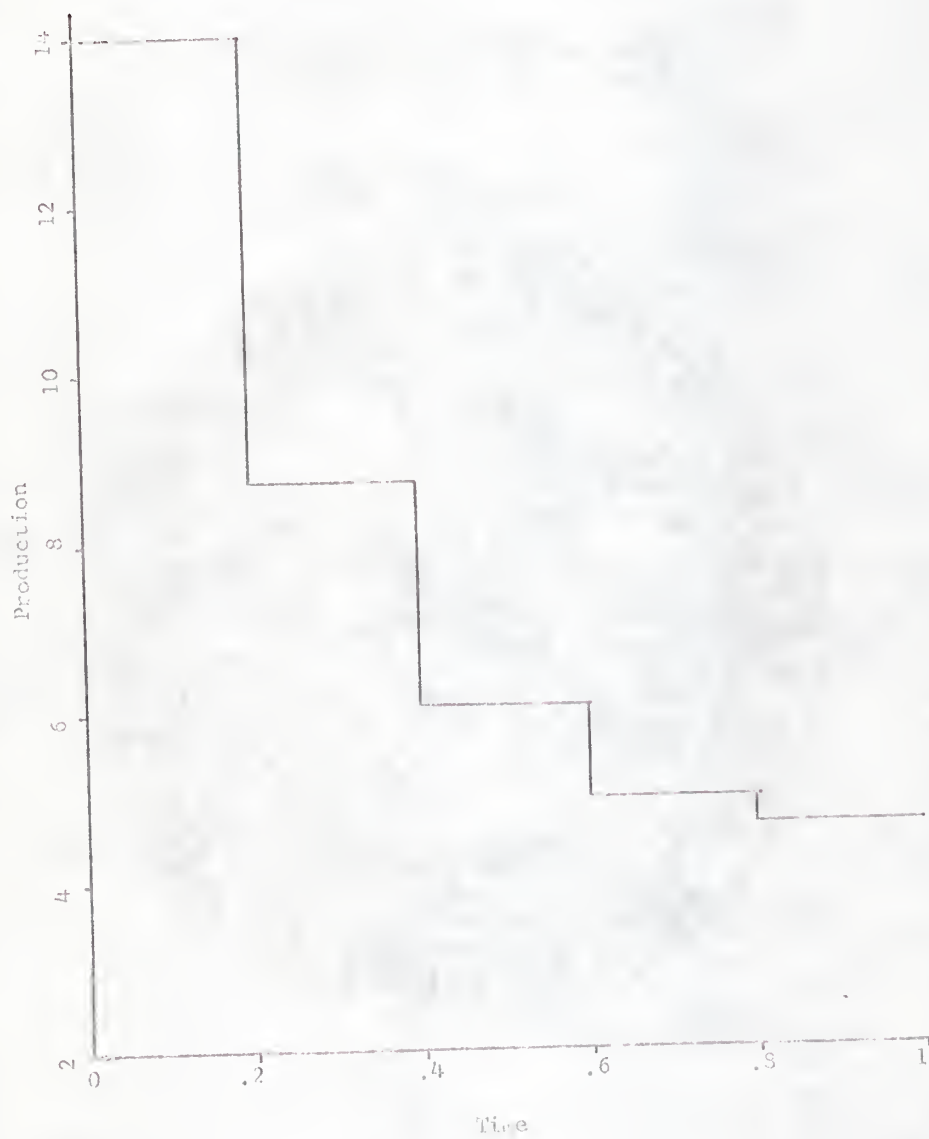


Fig. 3. Optimum production levels.

VI. DISCUSSION

The geometric programming algorithm in its present form can handle a large class of problems often found in practice.

In the posynomial case the method always produces a global minimum, not just a relative minimum. The minimum is equal to the maximum of the dual function whose constraints are linear. If the primal problem has zero degree of difficulty, the solution of the dual problem, hence the solution of primal problem, is obtained by solving a system of linear equations. In the case of zero degree of difficulty, each term in the optimal objective function has an invariant weight represented by the unique solution of the linear constraints. The mathematical importance of this property is that the weight of each term in the objective function is independent of the coefficients.

The extension of geometric programming, as developed by Wilde and Passy by using Kuhn-Tucker conditions, is applicable to any problem involving generalized polynomials. But any deviation from the full posynomial situation invalidates the arithmetic-geometric mean inequality and its useful applications. The optimal weights occur at stationary points of unspecified character in the general case, and this precludes direct search. Another difficulty in the general case is that one no longer has a guarantee that the solution obtained by working with the dual function corresponds to a minimum of the objective.

The existing theory of generalized geometric programming for polynomial optimization gives no way to compute optima

except in the special case where there is exactly one more term there are independent variables. Also the theory is formulated in terms of inequality constraints, although the physical restrictions occurring in practice are more often strict equalities. The Lagrangian algorithm for generalized polynomial optimization [3] is suitable for equality constrained problems, which was shown by rapid convergence for the production scheduling model with 23 degrees of freedom.

The convergence is not guaranteed for the above method. Moreover, even when the algorithm does converge, the point found may be a saddle point or even a maximum. Finally, a local minimum may not be the global minimum.

Sometimes during the initial iterations, the method takes too big a step and overshoots the minimum. This results in no convergence. This difficulty was overcome by restricting the step size to a predetermined percent of the variable.

Despite these difficulties, both geometric programming and the Lagrangian algorithm can be regarded as pioneering fields in nonlinear optimization with nonlinear constraints and they have great potentials in engineering design and systems analysis.

REFERENCES

1. Avriel, M. and D. J. Wilde, "Optimal Condenser Design by Geometric Programming," Ind. Eng. Chem. Process Des. Dev., 59, 256 (1967).
2. Beightler, C. S., R. M. Crisp and W. L. Meier, "Optimization by Geometric Programming," J. of Ind. Eng., 12, 117 (1968).
3. Blau, G. E. and D. J. Wilde, "A lagrangian algorithm for equality constrained generalized polynomial optimization," Symp. Optimization of reaction systems, Part 1, 65th Nat. meeting AIChE., 1969.
4. Duffin, R. J., "Dual Programs and Minimum Cost," J. SIAM., 10, 119 (1962).
5. Duffin, R. J., "Cost Minimization Problem Treated by Geometric Means," Operations Research, 10, 668 (1962).
6. Duffin, R. J., E. L. Peterson and C. Zener; Geometric Programming, John Wiley, New York (1966).
7. Frank, C. J., "Problem Solution Using the Geometric Programming Algorithm," Westinghouse Research Report 64-IHO-129-RI, (1964).
8. Frank, C. J., "Development of a Computer Algorithm for Geometric Programming," Westinghouse Research Report 64-IHO-129-R2, (1964).
9. Hadley, G., Linear Algebra, Addison-Wesley Publishing Co., Inc. (1961).
10. Hooke, R. and T. A. Jeeves, "Direct Search Solution of Numerical and Statistical Problems," J. Assoc. Comput. Mach., 8, (1961)
11. Lavi, A. and T. P. Vogl, Symposium on Recent Advances in Optimization Techniques, John Wiley, New York, (1966).
12. Passy, U. and D. J. Wilde, "Generalized Polynomial Optimization," J. SIAM, 15, 1744, (1967).
13. Passy, U. and D. J. Wilde, "A Geometric Programming Algorithm for Solving Chemical Equilibrium Problems," J. SIAM, 16, 363, (1968).
14. Scarborough, J. B., Numerical Mathematical Analysis, John Hopkins Press, Baltimore, (1950).

15. Sherwood, T. K., A Course in Process Design, MIT Press, Cambridge, (1963).
16. White, W. B., S. M. Johnson and G. B. Dantzig, "Chemical Equilibrium in Complex Mixtures," J. Chem. Phys., 28, 751 (1958).
17. Wilde, D. J., "A Review of Optimization Theory," Ind. Eng. Chem., 57, 18, (1965).
18. Wilde, D. J. and C. S. Beightler, Foundations of Optimization, Prentice Hall, N. J. (1967).
19. Zener, C., "A Mathematical Aid in Optimizing Engineering Designs," Proc. Nat. Acad. Sci. U.S.A., 47, 537, (1961).
20. Zener, C., "A Further Mathematical Aid in Optimizing Engineering Designs," Proc. Nat. Acad. Sci. U.S.A., 48, 518, (1962).
21. Zener, C., "Minimization of System Costs in Terms of Subsystem Costs," Proc. Nat. Acad. Sci. U.S.A., 51, 162, (1964).
22. Zener, C. and R. J. Duffin, "Optimization of Engineering Problems," Westinghouse Engineer, 154, (1964).

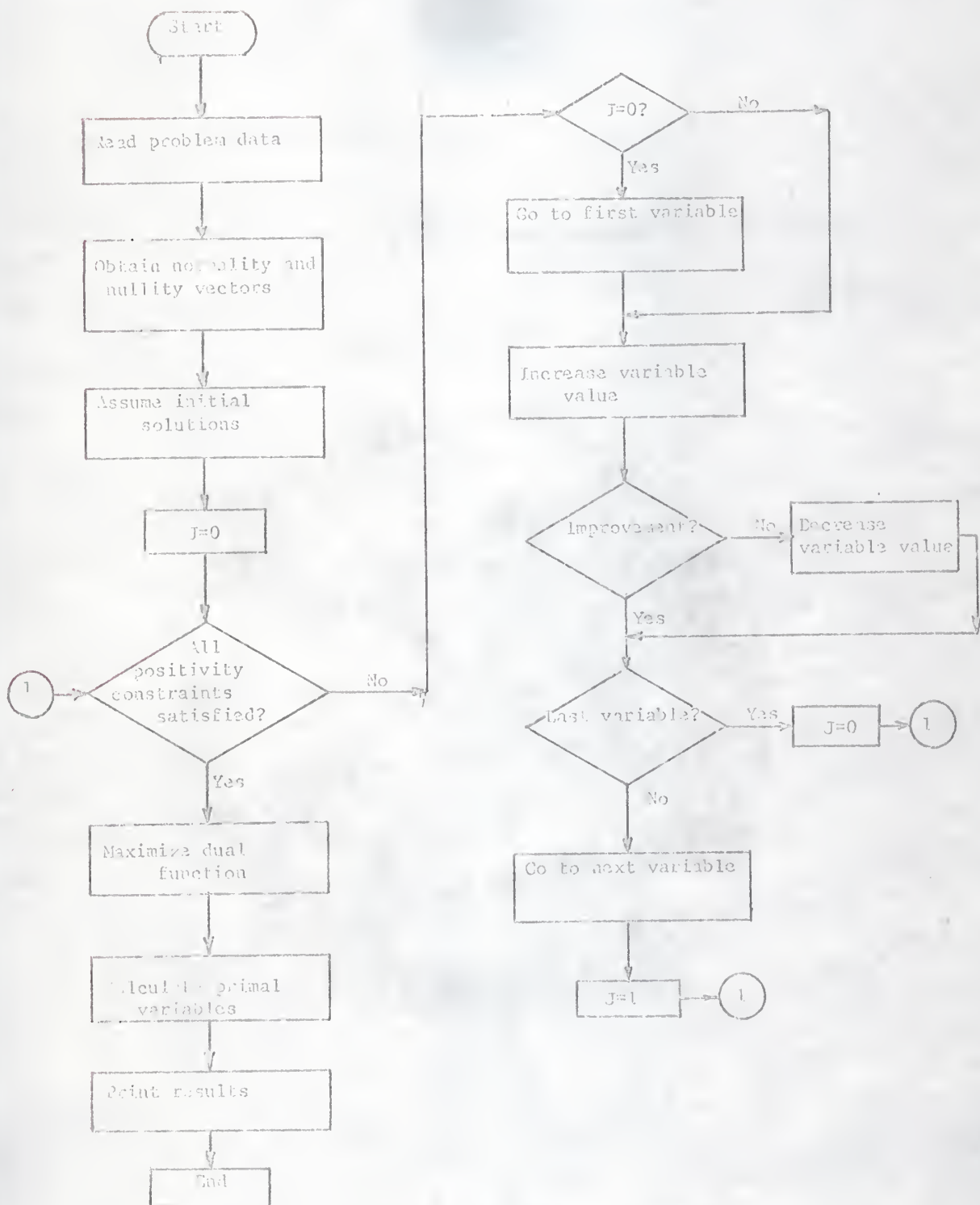


Fig. 9. Flow diagram for Geometric programming

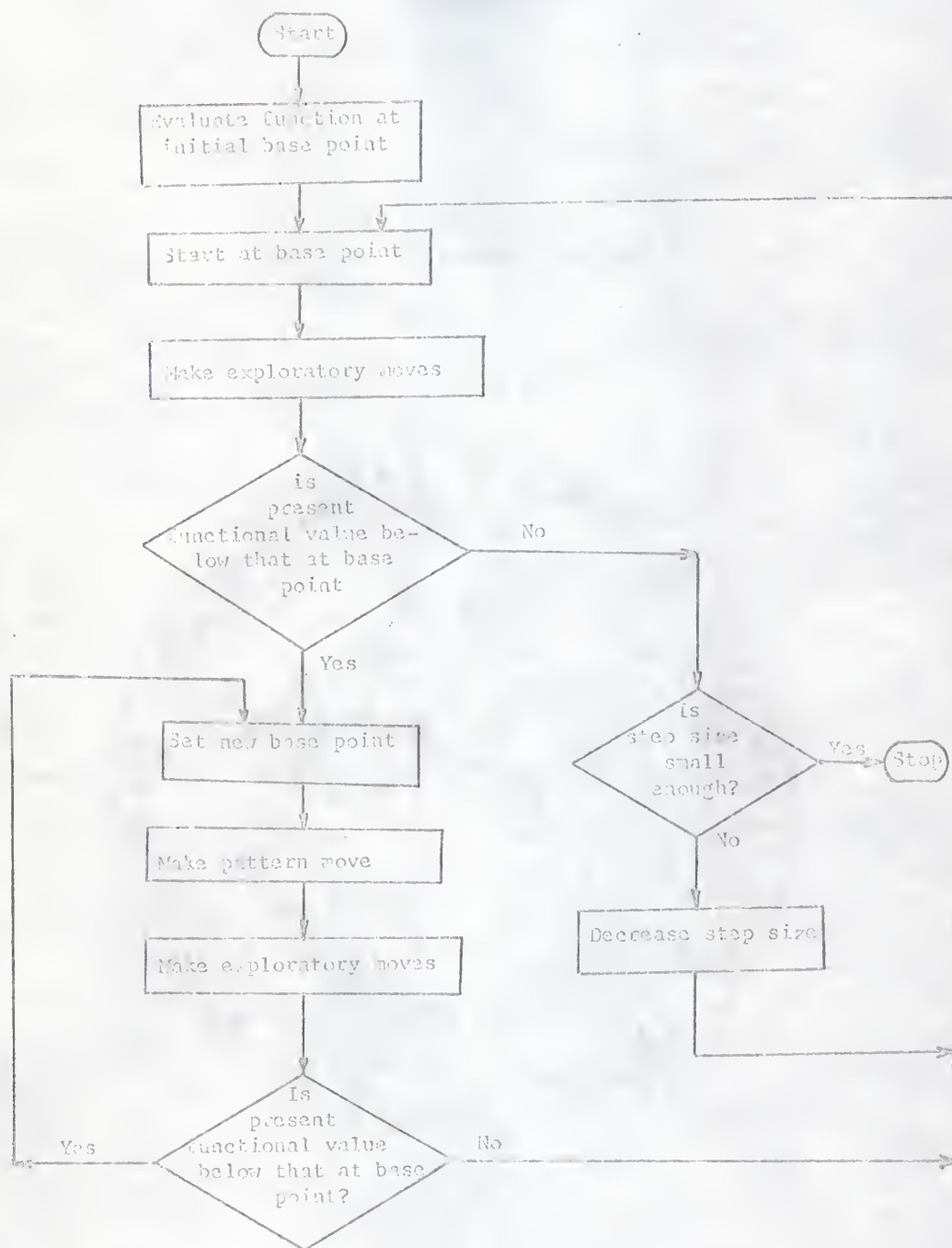


Fig. 10. Flow diagram for Hill and Jeeves pattern search. [10]

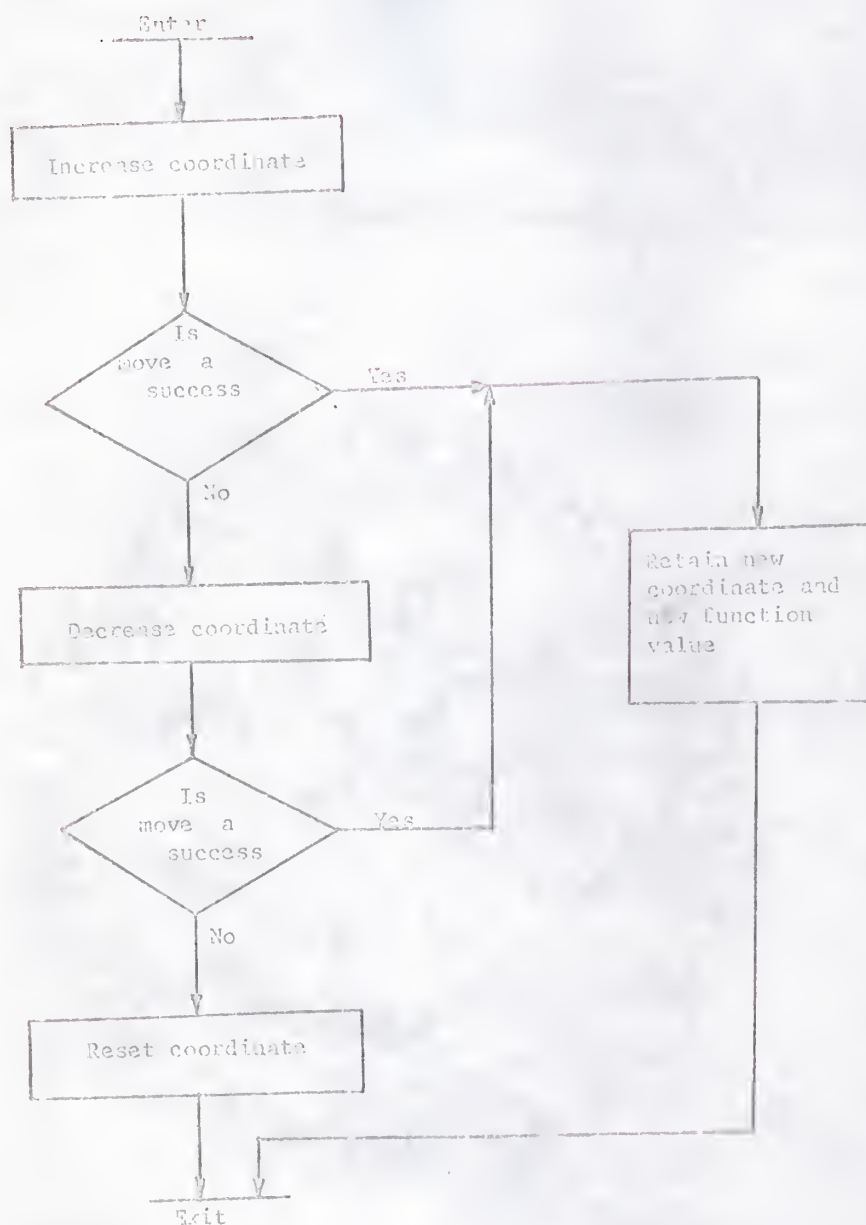


Fig. 11. Flow diagram for exploratory moves. [10]

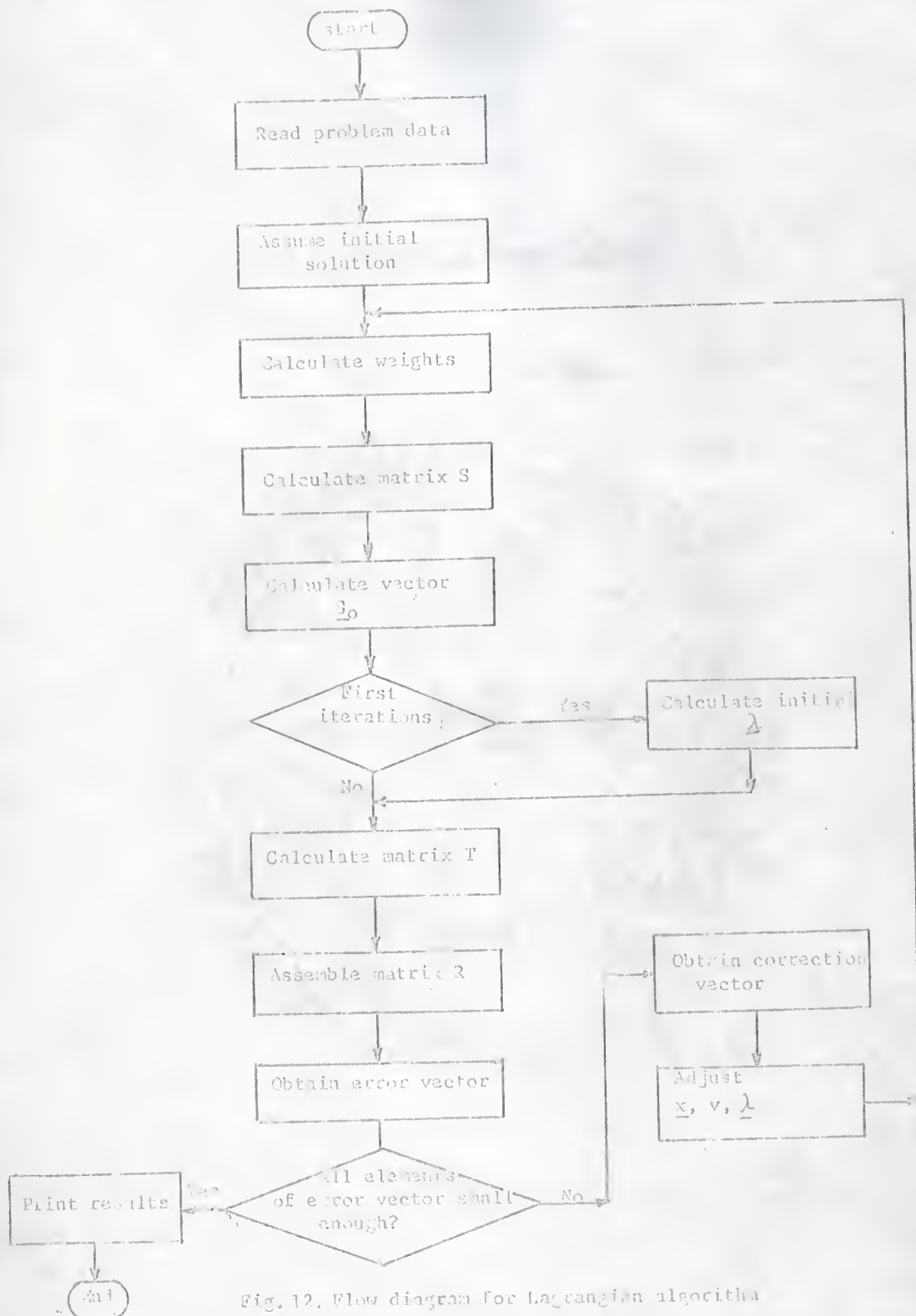


Fig. 12. Flow diagram for Lagrangian algorithm

APPENDIX B

C
C
C
C
C
C

THE COMPUTER PROGRAM FOR
THE SAMPLE PROBLEM BY GEOMETRIC PROGRAMMING
USING HOOKE AND JEEVES PATTERN SEARCH

```

1  DIMENSION A(50,25),AA(35,45),A1(30),A2(30),H(700),L2(30),L3(30),
   1  LG(50),ROLD(35), E(5),BASE1(35),BASE2(35),B1(50),B2(50),PA(5)
2  COMMON KA(20),KB(20),SUM(20),DEL(50),SIG(50),BET(20),ALAM(50),
   1  IC(50),R(35),S(50),B(50,35),NX,N3,K1,NM,KOUNT,KX
3  11 FORMAT (3I3)
4  11 FORMAT (14)
5  12 FORMAT (1H-, ' OPTIMUM ANNUAL COST',F25.2)
6  13 FORMAT (2F3.3)
7  18 FORMAT (' EXPONENT MATRIX SINGULAR')
8  22 FORMAT (F22.6)
9  READ 10,M,NX,K1
10 READ 11,(KA(I),I=1,K1)
11 READ 11,(KB(I),I=1,K1)
12 READ 13,((A(I,J),J=1,M),I=1,NX)
13 KOUNT=0
14 N1=NX-M-1
15 N2=M+1
16 N3=N1+1
17 N4=N2+1
18 NI=KA(1)-1
19 ND=KA(1)
20 DO 600 I=1,15
21 600 SIG(I)=1.
22 BET(1)=1.
23 BET(2)=1
24 C(1)=1000.
25 C(2)=4.*(10.)**9
26 C(3)=2.5*(10.)**5
27 C(4)=9000.
28 DO 100 I=1,NX
29 DO 100 J=1,M
30 100 AA(J+1,I)=A(I,J)*SIG(I)
31 DO 36 J=1,NM
32 36 AA(1,J)=SIG(J)
33 DO 101 J=ND,NX
34 101 AA(1,J)=0.
35 DO 98 I=1,N2
36 DO 98 J=1,N2
37 NA=N2*(J-1)
38 98 H(I+NA)=AA(I,J)
39 A1(1)=1.
40 DO 102 I=2,N2
41 102 A1(I)=1.
C  OBTAINING NORMALLITY AND NULLITY VECTORS
C  THE SUBROUTINES USED HERE ARE PROVIDED BY IBM
42 CALL MINV (H,N2,D,L2,L3)
43 IF(D.EQ.0.) GO TO 125
44 CALL GMPRD (H,A1,B1,N2,N2,1)
45 J1=1
46 DO 104 J=N4,NX
47 J1=J1+1
48 DO 103 I=1,N2
49 103 A2(I)=-AA(I,J)

```

```

52 CALL G4PRD (H,A2,B2,I2,I2,1)
51 DO 104 I=1,N2
52 104 B(I,J1)=B2(I)
53 DO 50 I=1,N2
54 50 B(I,1)=B1(I)
55 I1=0
56 DO 51 I=N4,NX
57 I1=I1+1
58 DO 51 J=1,N3
59 IF(J-I1-1) 52,53,52
60 53 B(I,J)=1.
61 GO TO 51
62 C(I,J)=0.
63 51 CONTINUE
64 DO 5 I=1,N1
65 5 R(I+1)=.8
C OBTAINING INITIAL FEASIBLE SOLUTION
66 I1=1
67 ITER=0
68 NTER=0
69 126 DO 128 J=2,N3
70 128 ROLD(J)=R(J)
71 127 L1=1
72 J1=2
73 NC=1
74 200 DO 111 I=1,NX
75 S(I)=0.
76 DO 110 J=2,N3
77 11 S(I)=S(I)+R(J)*B(I,J)
78 D=LT(I)=R(I,1)+S(I)
79 111 CONTINUE
80 IF(ITER-1000) 210,129,129
81 21 IF(DEL(I1)) 206,207,207
82 206 CH(I1)=DEL(I1)
83 IF(NC) 217,217,218
84 217 IF(CH(I1)-OLD) 218,209,209
85 218 GO TO (310,211),L1
86 310 R(J1)=ROLD(J1)-.1
87 ITER=ITER+1
88 NC=0
89 OLD=CH(I1)
90 L1=2
91 GO TO 200
92 211 R(J1)=ROLD(J1)+.1
93 ITER=ITER+1
94 GO TO 200
95 209 OLD=CH(I1)
96 ROLD(J1)=R(J1)
97 L1=1
98 NC=1
99 IF(J1-N3) 214,220,200
100 214 J1=J1+1
101 GO TO 200
102 220 J1=2
103 GO TO 200
104 207 I1=1
105 ROLD(J1)=R(J1)
106 212 IF(DEL(I1)) 127,213,213
107 213 IF(I1-NX) 215,216,216
108 215 I1=I1+1

```

```

109      GO TO 212
110      216 CONTINUE
C      HOOKE AND JEEVES PATTERN SEARCH
111      DE=.1
112      DO 189 L=2,N3
113      189 BASE1(L)=R(L)
114      191 CALL FUNC(V)
115      F=V
116      L=1
117      N=1
118      191 DO 307 J=2,N3
119      R(J)=BASE1(J)+DE
120      CALL FUNC(V)
121      GO TO (301,302),N
122      301 IF(V-T) 303,303,304
123      302 IF(V-ALD) 305,305,304
124      303 ALD=T
125      315 R(J)=BASE1(J)-DE
126      CALL FUNC(V)
127      IF(V-ALD) 306,306,304
128      306 ROLD(J)=BASE1(J)
129      R(J)=BASE1(J)
130      GO TO 307
131      304 ROLD(J)=R(J)
132      ALD=F
133      307 N=2
134      GO TO (312,313),L
135      313 IF(ALD-VMAX) 315,315,309
136      312 IF(ALD-T) 308,308,309
137      308 IF(DE-.001) 410,410,311
138      311 DE=DE/2.
139      DO 528 J=2,N3
140      528 R(J)=ROLD(J)
141      GO TO 191
142      309 VMAX=ALD
143      DO 314 J=2,N3
144      BASE2(J)=ROLD(J)
145      314 BASE1(J)=2.*BASE2(J)-BASE1(J)
146      L=2
147      DO 529 J=2,N3
148      529 R(J)=BASE1(J)
149      NTER=NTER+1
150      IF(NTER-1000) 191,191,129
151      315 DO 316 J=2,N3
152      316 BASE1(J)=BASE2(J)
153      DO 130 J=2,N3
154      130 R(J)=BASE1(J)
155      NTER=NTER+1
156      IF(NTER-1000) 100,100,129
157      410 DO 317 J=2,N3
158      317 R(J)=BASE1(J)
159      VD=F
160      PRINT 12,VD
C      OBTAINING OPTIMUM PRIMAL VARIABLES
161      X1=DEL(1)*VD/C(1)
162      X2=DEL(3)/(ALAM(1)*C(3))
163      PRINT 22,X1
164      PRINT 22,X2
165      GO TO 119
166      129 CONTINUE

```



```

171 SUBROUTINE FUNC(V)
172 COMMON KA(20),KB(20),SUM(20),DEL(50),SIG(50),BET(20),ALAM(50),
173     IC(50),R(35),S(50),L(50,35),NX,N3,K1,NN,KOUNT,KX
22  FORMAT (F22.6,I5,F8.3)
    SIGMA=1.
    KOUNT=KOUNT+1
    DO 111 I=1,NX
    S(I)=0.
    DO 110 J=2,N3
110  S(I)=S(I)+R(J)*P(I,J)
111  DEL(I)=B(I,1)+S(I)
    I1=1
121  IF(DEL(I1)) 117,117,118
118  IF(I1-NX)119,120,120
119  I1=I1+1
    GO TO 121
117  V=-999
    RETURN
120  DO 108 K=1,K1
    L7=KA(K)
    L8=KB(K)
    SUM(K)=0.
    DO 20 I=L7,L8
20  SUM(K)=SUM(K)+SIG(I)*DEL(I)
    ALAM(K)=BET(K)*SUM(K)
    IF(ALAM(K).LT.0.) GO TO 117
108  CONTINUE
    P1=1.
    DO 112 I=1,NN
    IF(DEL(I).EQ.0.) GO TO 112
    P1=P1*(C(I)/DEL(I))**((SIG(I)*DEL(I)))
112  CONTINUE
    P2=1.
    DO 114 K=1,K1
    L7=KA(K)
    L8=KB(K)
    DO 114 I=L7,L8
    IF(DEL(I).EQ.0.) GO TO 114
    P2=P2*(C(I)*ALAM(K)/DEL(I))**((SIG(I)*DEL(I)))
114  CONTINUE
    V=SIGMA*(P1*P2)**SIGMA
    PRINT 22,V,KOUNT,R(2)
    RETURN
    END
213

```

```

      GEOMETRIC, SUN, C, CK, TIME=C, PAGES=50
%JOB
C   COMPUTER PROGRAM FOR SEA, OVER PROBLEM BY GEOMETRIC PROGRAMMING
C   USING THE NEWTON RAPHSON PROCEDURE
1   DIMENSION C(50),A(50,25),AA(25,25),AL(25),A2(25),H(700) B(50,30),
      IB1(50),B2(50),AK(30),DLAM(30,30),R(30),S(40),SS(30,30),LA(30),
      ISB(30,30),AE(30,30),SC(30),SD(30),F(30),AF(30,30),F1(30),EPS(30),
      IDEL(50),ALAM(30),L2(30),L3(30),CH(50),ROLD(30),I(5),E(5),SIG(50),
      LBET(20),KA(20),KB(20),SUN(20)
2   10 FORMAT (3I3)
3   11 FURMAT (4I4)
4   12 FORMAT (5I2,3)
5   13 FORMAT (7F3.0)
6   14 FORMAT (15)
7   15 FORMAT (7F8.3)
8   18 FORMAT ('      EXPONENT MATRIX SINGULAR')
9   READ 10,M,N,K1
10  READ 11,(KA(I),I=1,K1)
11  READ 11,(KB(I),I=1,K1)
12  READ 13,((A(I,J),J=1,M),I=1,N)
13  N1=N-M-1
14  N2=M+1
15  N3=N1+1
16  N4=N2+1
17  C(1)=1.
18  C(2)=40.
19  C(3)=0.18
20  C(4)=44.5
21  C(5)=0.000000006
22  C(6)=0.0000000215
23  C(7)=1.
24  C(8)=1.
25  C(9)=1.
26  C(10)=1.
27  DO 44 I=1,10
28  44 SIG(I)=1.
29  DO 30 I=1,4
30  30 BET(I)=1.
31  DO 99 I=1,N1
32  99 F1(I)=0.
33  DO 100 I=1,N
34  DO 100 J=1,N
35  100 AA(J+1,I)=A(I,J)
36  AA(1,1)=1.
37  DO 101 J=2,N
38  101 AA(1,J)=0.
39  DO 98 I=1,N2
40  DO 98 J=1,N2
41  NA=N2*(J-1)
42  98 H(I+NA)=AA(I,J)
43  A1(1)=1.
44  DO 102 I=2,N2
45  102 A1(I)=0.
C   OBTAINING THE NORMALITY AND NULLITY VECTORS
C   THE SUBROUTINES USED HERE ARE PROVIDED BY IDH
46  CALL MINV (H,N2,D,L2,L3)
47  IF(D.EQ.0.) GO TO 125
48  CALL GMPRO (H,A1,B1,N2,N2,1)
49  J1=1
50  DO 104 J=N4,N
51  J1=J1+1

```

```

52      DO 103 I=1,N2
53 103  A2(I)=-A1(I,J)
54      CALL GMRD (H,A2,D2,N2,N2,1)
55      DO 104 I=1,N2
56 104  B(I,J)=B2(I)
57      DO 50 I=1,N2
58      50  B(I,1)=B1(I)
59          I1=0
60          DO 51 I=N4,N
61          I1=(I+1)
62          DO 51 J=1,N3
63          IF(J-I1-1) 52,53,52
64      53  B(I,J)=1.
65          GO TO 51
66      52  B(I,J)=0.
67      51  CONTINUE
68          DO 105 J=1,N3
69          AK(J)=1.
70          DO 105 I=1,N
71          AK1=C(I)**(B(I,J)*SIG(I))
72 105  AK(J)=AK(J)*AK1
73          DO 108 I=1,K1
74          L7=KA(I)
75          L8=KB(I)
76          DO 108 J=1,N3
77          SUM(I)=0.
78          DO 20 K=L7,L8
79      20  SUM(I)=SUM(I)+B(K,J)
80          DLAM(I,J)=SUM(I)
81 108  CONTINUE
82          DO 5 I=1,M1
83      5  R(I+1)=.4
C      OBTAINING INITIAL FEASIBLE SOLUTION
84          I1=1
85          ITER=0
86          NTER=0
87 126 DO 128 J=2,N3
88 128  ROLD(J)=R(J)
89 127  L1=1
90          J1=2
91          NC=1
92 200 DO 111 I=1,N
93          S(I)=0.
94          DO 110 J=2,N3
95 110  S(I)=S(I)+R(J)*B(I,J)
96          DEL(I)=B(I,1)*S(I)
97 111  CONTINUE
98          IF(ITER-100) 210,210,129
99 210  IF(DEL(I1)) 206,206,207
100 206  CH(I1)=DEL(I1)
101          IF(NC) 217,217,210
102 217  IF(CH(I1)-OLD) 218,209,209
103 218  GO TO (310,211),L1
104 310  R(J1)=ROLD(J1)+.05
105          ITER=ITER+1
106          NC=0
107          OLD=CH(I1)
108          L1=2
109          GO TO 200
110 211  R(J1)=ROLD(J1)+.05

```

```

111      ITER=ITER+1
112      GO TO 200
113 209 OLD=CH(I1)
114      ROLD(J1)=R(J1)
115      LI=1
116      MC=1
117      IF(J1-N3) 216,220,200
118 214 J1=J1+1
119      GO TO 200
120 220 J1=2
121      GO TO 200
122 207 I1=1
123      ROLD(J1)=R(J1)
124 212 IF(DEI(I1)) 127,127,213
125 213 IF(I1-N) 215,215,216
126 215 I1=I1+1
127      GO TO 212
128 216 DO 112 K=1,K1
129      SA(K)=0.
130      DO 113 J=2,N3
131 113 SA(K)=SA(K)+R(J)*DLAM(K,J)
132 112 ALAM(K)=DLAM(K,1)+SA(K)
133      DO 121 L=2,N3
134      DO 121 K=2,N3
135      SS(L,K)=0.
136      DO 121 I=1,N
137 121 SS(L,K)=SS(L,K)+SIG(I)*B(I,L)*B(I,K)/DEI(I)
138      DO 122 I=2,N3
139      DO 122 L=2,N3
140      SB(L,I)=0.
141      DO 122 K=1,K1
142 122 SB(L,I)=SB(L,I)+DET(K)*DLAM(K,L)*DLAM(K,I)/ALAM(K)
143      DO 114 I=2,N3
144      DO 114 J=2,N3
145 114 AE(I-1,J-1)=SS(I,J)-SB(I,J)
146      DO 118 J=2,N3
147      SC(J)=0.
148      SD(J)=0.
149      DO 115 I=1,N
150 115 SC(J)=SC(J)+SIG(I)*B(I,J)*ALOG(DEI(I))
151      DO 117 K=1,K1
152 117 SD(J)=SD(J)+DET(K)*DLAM(I,J)*ALOG(ALAM(K))
153 118 F(J-1)=SC(J)-SD(J)-ALOG(AK(J))
154      GO TO 201
155 204 DO 116 I=1,N1
156      DO 116 J=1,N1
157 116 AF(I,J)=AE(I,J)
C      NEWTON RAPHSON PROCEDURE FOR SOLUTION OF SIMULTANEOUS EQUATIONS
158      DO 500 J=1,N1
159 500 F1(I)=F(J)
160      CALL DETR(AE,N1,D1)
161      DO 55 I=1,N1
162      DO 56 J=1,N1
163 56 AF(J,I)=-F(J)
164      CALL DETR(AF,N1,D2)
165      E1=D2/D1
166      R(I+1)=R(I+1)+E1
167      DO 57 J=1,N1
168 57 AF(J,I)=AE(J,I)
169 55 CONTINUE

```



```

170      NTER=ITER+1
171      IF(NTER-50) 126,126,129
172      DO 181 I=1,N1
173      130 EPS(I)=ABS(FL(I)-F(I))
174      I=1
175      202 IF(EPS(I)+.01) 203,203,204
176      203 I=I+1
177      IF(I-N1) 202,202,205
178      205 P=1.
179      DO 123 I=1,N
180      123 P=P*(C(I)/DEL(I))*(SIG(I)*DEL(I))
181      P1=1.
182      DO 124 K=1,K1
183      124 P1=P1*ALAM(K)*(ALAM(K)*BE(K))
184      VD=P*P1
185      PRINT 12,VD
C      OBTAINING OPTIMUM PRIMAL VARIABLES
186      AY=DEL(1)*VD/C(1)
187      ALF=DEL(7)/(ALAM(4)*C(7))
188      BE =DEL(8)/(ALAM(4)*C(8))
189      BETD=DEL(9)/(ALAM(4)*C(9))
190      GAM=DEL(10)/(ALAM(4)*C(10))
191      Q=C(4)*ALAM(3)/(ALF*DEL(4))
192      U=Q*ALAM(1)/(AY*BE *DEL(2))
193      PRINT 15,AY,ALF,BE,BETD,GAM,Q,U
194      GO TO 119
195      129 PRINT 14, NTER
196      PRINT 14, ITER
197      GO TO 119
198      125 PRINT 18
199      119 STOP
200      END

```



```

201 SUBROUTINE DETP(A,K,DET)
202 DIMENSION A(6,6)
203 1 Z=1.0
204 DO 9 I=2,K
205 IF (A(I-1,I-1))3,4,3
206 4 DO 5 I1=M,K
207 IF(A(I-1,I1))6,7,6
208 7 DET=0.
209 5 CONTINUE
210 RETURN
211 6 I3=I-1
212 DO 8 I2=I3,K
213 TEMP=A(I2,I3)
214 A(I2,I3)=A(I2,I1)
215 8 A(I2,I1)=TEMP
216 Z=Z*(-1.)
217 3 DO 9 I=M,K
218 R=A(I,M-1)/A(M-1,M-1)
219 DO 9 J=M,K
220 9 A(I,J)=A(I,J)-A(M-1,J)*R
221 DET=1.0
222 DO 18 I=1,K
223 18 DET=DET*A(I,I)
224 DET=DET*Z
225 RETURN
226 END

```

JOB

DT=C*DTK, TIME=3, PAGES=50

```

C  COMPUTER PROGRAM FOR THE PRODUCTION SCHEDULING PROBLEM
C  BY THE LAGRANGIAN ALGORITHM USING 50 PERCENT FORCING
1  DIMENSION KA(6),X(10),SIG(11,21),A(10,21,10),W(10,20),G(10),S(15,
    15),SB(15),SC(150),L6(25),L7(25),R5(150),R6(150),R7(15),ALAN(15),
    LWB(40),T(15,15),TB(250),R(25,25),TEM(15),E(20),RW(700),COR(50),Q(5
    1),C(8,20),JUT(30)
2  11 FORMAT (2I3)
3  11 FORMAT (6I3)
4  12 FORMAT (10F3.0)
5  13 FORMAT (5F5.2)
6  15 FORMAT (F15.6)
7  16 FORMAT (10F10.4)
8  17 FORMAT (10F12.6)
9  18 FORMAT (5F8.2)
10 19 FORMAT (16F8.2)
11 20 FORMAT (1H-, '      ITERATION NUMBER',15)
12 21 FORMAT (6F10.3)
13 22 FORMAT (10F10.2)
14 23 FORMAT (F15.6)
15 READ 10,MC,NC
16 READ 11,(KA(I),I=1,NC)
17 DO 1 I=1,NC
18   LI=KA(I)
19   DO 61 J=1,LI
20     READ 12,(A(I,J,K),K=1,NC)
21 61 PRINT 12,(A(I,J,K),K=1,NC)
22 READ 13,C1,C2,C3,C4,DT
23 Q(1)=2.10*DT
24 Q(2)=2.30*DT
25 Q(3)=2.50*DT
26 Q(4)=2.70*DT
27 Q(5)=2.90*DT
28 DO 50 J=1,5
29 51 C(1,J)=C1*DT
30 DO 51 J=11,15
31 51 C(1,J)=2.*C1*C2*DT
32 DO 53 J=6,10
33 53 C(1,J)=C3*DT
34 DO 52 J=16,20
35 52 C(1,J)=2.*C3*C4*DT
36 C(2,1)=1/(5-Q(1))
37 C(2,2)=DT/(5-Q(1))
38 C(3,1)=1/Q(2)
39 C(3,2)=1/Q(2)
40 C(3,3)=DT/Q(2)
41 C(4,1)=1/Q(3)
42 C(4,2)=1/Q(3)
43 C(4,3)=DT/Q(3)
44 C(5,1)=1/Q(4)
45 C(5,2)=1/Q(4)
46 C(5,3)=DT/Q(4)
47 C(6,1)=1/Q(5)
48 C(6,2)=1/Q(5)
49 C(6,3)=DT/Q(5)
50 CONS=C1*C2**2+C3*C4**2
51 DO 56 K=1,10
52 56 SIG(1,K)=1.
53 DO 57 K=11,20
54 57 SIG(1,K)=-1.

```

```

5      DO 60 M=2,MC
6      L1=KA(M)
7      DO 60 K=1,L1
8      SIG(M,K)=1.
9      SIG(2,2)=-1.
10     SIG(3,1)=-1.
11     SIG(4,1)=-1.
12     SIG(5,1)=-1.
13     SIG(6,1)=-1.
14     DO 62 I=1,10
15     62 X(I)=5.0
16     I=0
17     DO 140 M=1,MC
18     SUM=0.
19     L1=KA(M)
20     DO 141 K=1,L1
21     P=1.
22     DO 142 N=1,NC
23     142 P=P*(X(I)**A(M,K,N))
24     141 SUM=SUM+SIG(M,K)*C(M,K)*P
25     140 G(M)=SUM
26     IF(I.GT.1) GO TO 138
27     V=G(1)
28     L1=KA(1)
29     DO 100 K=1,L1
30     P=1.
31     DO 101 N=1,NC
32     101 P=P*(X(N)**A(1,K,N))
33     100 V(1,K)=C(1,K)*P/G(1)
34     138 DO 102 M=2,MC
35     L1=KA(M)
36     DO 102 K=1,L1
37     P=1.
38     DO 103 N=1,NC
39     103 P=P*(X(N)**A(M,K,N))
40     102 W(M,K)=C(M,K)*P
41     DO 105 N=1,MC
42     DO 105 M=1,MC
43     P2=0.
44     L2=KA(M)
45     DO 104 K=1,L2
46     104 P2=P2+SIG(M,K)*A(M,K,N)*W(M,K)
47     105 S(N,M)=P2
48     DO 106 N=1,NC
49     106 SB(N)=S(N,1)
50     KX=0
51     MD=MC-1
52     DO 107 M=2,MC
53     DO 107 N=1,NC
54     KX=KX+1
55     107 SC(KX)=S(N,M)
56     CALL CMTRA (SC,R5,NC,MD)
57     CALL GMPRD (R5,SC,OUT,MD,NC,MD)
58     CALL MINV (OUT,MD,0,L6,L7)
59     CALL GMPRD (OUT,R5,R6,MD,MD,NC)
60     CALL GMPRD (R6,SB ,R7,MD,NC,1)
61     DO 410 J=1,MD
62     410 ALAP(J)=R7(J)
63     DO 110 N=1,NC
64     DO 110 J=1,NC

```

```

115      S1=0.
116      S2=0.
117      S3=0
118      L1=KA(1)
119      DO 111 K=1,L1
120 111  S1=S1+SIG(1,K)*A(1,K,M)*A(1,K,J)*W(1,K)
121      DO 112 M=2,MC
122      L2=KA(M)
123      DO 113 K=1,L2
124 113  S2=S2+SIG(M,K)*A(M,K,N)*A(M,K,J)*W(M,K)
125 112  S3=S3+ALAM(M-1)*S2
126 111  T(1,J)=S3-S1
127      K3=NC+MC
128      K=0
129      N=4C
130      DO 115 J=1,K3
131      K1=1
132      DO 115 K=1,K3
133      IF(J-N) 116,116,118
134 116  IF(K-N) 117,117,124
135 117  R(J,K)=T(J,K)
136      GO TO 115
137 118  IF (J-N-1) 119,119,127
138 119  IF(K-N) 120,120,121
139 120  R(J,K)=SB(K)
140      GO TO 115
141 121  IF(K-N-1) 122,122,123
142 122  R(J,K)=-1
143      GO TO 115
144 123  R(J,K)=0.
145      GO TO 115
146 124  IF(K-N-1) 125,125,126
147 125  R(J,K)=SB(J)
148      GO TO 115
149 126  K1=K1+1
150      R(J,K)=S(J,K1)
151      GO TO 115
152 127  IF(K-N) 128,128,129
153 128  K2=K2+1
154      R(J,K)=SC(K2)
155      GO TO 115
156 129  R(J,K)=0.
157 115  CONTINUE
158      SIGM=G(1)/ABS(G(1))
159      CALL GMPRO (SC,ALAM,TEM,NC,ND,1)
160      DO 130 J=1,NC
161 130  E(J)=SB(J)-TEM(J)
162      E(NC+1)=1-G(1)/V
163      J2=1
164      KE=NC+2
165      DO 131 J=KE,K3
166      J2=J2+1
167 131  E(J)=1-G(J2)
168      J=1
169 503 IF(ABS(E(J))-0.0001) 501,501,502
170 501 J=J+1
171      IF(J-K3) 503,503,504
172 502 J2=0
173      DO 132 K=1,K3
174 132 J=1,K3

```

```

177      J2=J2+1
178      132  W(J2)=R(J,K)
179      CALL EINV (RW,K3,D,LA,17)
180      CALL GMPD (RW,E,COR,K3,K3,1)
181      PRINT 20,I
182      PRINT 17,(X(N),N=1,NC)
183      PRINT 19,(E(N),N=1,K3)
184      PRINT 15,G(1)
185      IA=0
186      DO 133 N=1,NC
187      WB(N)=X(N)
188      133  X(N)=X(N)*EXP(COR(N))
189      PRINT 17,(X(N),N=1,NC)
190      DO 150 N=1,NC
191      CH=ABS(X(N)-WB(N))
192      CHI=X(N)-WB(N)
193      IF(CH.GT.(.5*WB(N))) GO TO 1002
194      GO TO 150
195      1002  AMU=CHI/CH
196      X(N)=WB(N)+.5*AMU*WB(N)
197      IA=1
198      150  CONTINUE
199      IF(IA.GT.0) GO TO 1001
200      V=V*EXP(COR(NC+1))
201      J2=0
202      ND=NC+2
203      DO 134 M=ND,K3
204      J2=J2+1
205      134  ALAM(J2)=ALAM(J2)+COR(M)
206      LL=KA(1)
207      DO 135 K=1,LL
208      P=1.
209      DO 136 N=1,NC
210      136  P=P*X(N)**A(1,K,N)
211      135  W(1,K)=C(1,K)*P/V
212      140  I=I+1
213      GO TO 500
214      500  G(1)=G(1)+CONS
215      PRINT 15,G(1)
216      PRINT 17,(X(N),N=1,NC)
217      STOP
218      END

```


APPLICATION OF GEOMETRIC PROGRAMMING
TO INDUSTRIAL SYSTEMS

by

NAYAN BHATTACHARYA

B. Tech. (Hons.), Mechanical, Indian Institute of Technology
Kharagpur, India, 1966

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1969

Most of the optimization problems which occur in real-life systems are nonlinear in nature with nonlinear constraints. Conventional optimization techniques can solve these problems only after linearizing the constraints and hence at the sacrifice of the accuracy. Geometric programming is a recently developed technique which can handle very efficiently a subclass of the above problems characterized by functions as polynomials with positive coefficients. Wilde's extension of geometric programming makes geometric programming applicable to even a broader class in which the functions are expressed as generalized polynomials. But difficulty in numerical analysis often restricts the use of extended geometric programming. Wilde's Lagrangian algorithm is useful in cases of generalized polynomials with equality constraints.

The purpose of this thesis is to apply these recently developed techniques to different engineering and industrial management systems and to make a critical analysis of the results and computational procedure used.

First a brief review of geometric programming and its extension is made, and computational procedure is discussed. A review of the Lagrangian algorithm follows. Then various approximation techniques which were applied to transform a general problem to the required form of geometric programming are discussed. Finally six problems are solved. The first problem is for illustration purposes, while the next four are engineering design problems with different degrees of complexity. The last is a production scheduling problem which is solved by the

Lagrangian algorithm.

The advantages as well as disadvantages of geometric programming and the Lagrangian algorithm are highlighted. A modification of the Lagrangian algorithm has been suggested.